SOME FINDINGS RELATED TO SIMPLE AND CAPACITATED PLANT LOCATION PROBLEMS

A Thesis Submitted
In Partial Fulfillment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

February, 2003

by

RAVINDRA GUPTA

DEPARTMENT OF INDUSTRIAL &MANAGEMENT ENGINEERING
INDIAN INSTITUTE O F TECHNOLOGY
KANPUR - 208016 (INDIA)

CERTIFICATE

It is certified that the work contained in the thesis entitled, *"Some Findings Related To Simple and Capacitated Plant Location Problems"* by Ravindra Gupta has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

(Dr. RRK Sharma)

February 2003

Professor
Industrial and Management Engineering
Indian Institute of Technology, Kanpur.

ACKNOWLEDGEMENT

This is my privilege to express thanks to a long list of people and unseen hands that have made contribution with some acknowledgeable views.

It really puts me in euphoria to express my deep sense of gratitude to my thesis guide Dr. R.R.K. Sharma, for guiding me throughout my work whenever I needed his help, as well as for his infallible suggestions, scintillating discussions, constant encouragement and constructive criticism.

It has been a wonderful learning experience to me at IITK. I wish to express my sincere thanks to all the faculty members of IME department who enlightened us with their knowledge during this one and half year period. Finally, I am thankful to all our classmates and friends Arun Pratap Singh, Santosh and Ashish who made my stay at IITK unforgettable.

Finally I thank my parents for their support and encouragement.

ABSTRACT

In this thesis we concluded an empirical investigation to find that the new formulation for SPLP, as developed by Muralidhar (2000), was in fact a weak one. Later we solved a global set of problems of various sizes of SPLP to find that the Dual Ascent procedure due to Erlenkotter (1978) produces solutions that were within 8% of the optimal solution to SRS.

Berry (2003) had added new "Strong Relaxation" type constraints to SSCWLP (Single Stage Capacitated Warehouse Location Problem) and found they were effective. We added similar constraints to SPLP and found that they were ineffective. However when these constraints were added to Capacitated Plant Location Problems (CPLP), they were highly effective as indicated by a numerical study.

CONTENTS

APPENDICES

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Plant location problem is to choose a sufficient number of plants, their capacity may be limited or unlimited (each location has an associated fixed cost) so that a given number of markets (each with a known demand) can be serviced while minimizing the sum of fixed costs of plant location and the transportation costs of shipping goods from plants to markets (cost of unit shipment being fixed for a given pair of plant and market). The limited capacity problem is known as CPLP (Capacitated Plant Location Problem) and the problem with unlimited capacity is termed as SPLP (Simple Plant Location Problem)/ Uncapacitated Plant Location Problem. Both the location problems SPLP and CPLP have been extensively studied in literature.

In this work, we explore the previous work of Erlenkotter [9] on SPLP, and tried to find the efficiency of Dual Ascent procedure to solve SPLP, and then we have tried the previous work of Sharma and Muralidhar [18]. We have tried a new formulation of CPLP at here. The new formulation is giving better bounds in comparison to SRS of CPLP.

The organization of thesis is as follows:

In chapter two, we give relevant literature review. In chapter three, we have explained the research problems in detail. In chapter four, we have discussed about the efficacy of Dual Ascent procedure. In chapter five, we have given the comparison of different formulations, in this we discussed the WRS (Old/New) and SRS formulation and compared them. In chapter six we have included Big 'M' constraint to all the formulations and tried to find its performance over the older ones. In chapter seven, we have given a new formulation for problem CPLP and have shown that the new formulation is giving better bounds to the SRS of CPLP. Chapter eight encompasses the results and conclusions on our study. It covers the directions in which future studies and research may be done.

We have solved various SPLP and CPLP problems by using Lingo software and by the program written in 'C' for Dual Ascent procedure. All the lingo formulations and the code written for Dual Ascent procedure is given in Appendix.

CHAPTER 2
LITERATURE REVIEW

2.1 Introduction:

This chapter presents comprehensive analysis of the research work done in the area of study of various relaxations for the SPLP. It also focuses upon the relative strengths of various relaxations so far being studied. We start with an extensive discourse on solution properties and computational techniques spanning from early heuristics to presumably most novel exact methods. Various formulations originating from different LP-relaxations given by researchers for SPLP and their exact algorithms were discussed further. Then dual based solution procedure given by Erlenkotter [9], and finally a new relaxation of SPLP given by Sharma and Muralidhar [18] has been discussed.

2.2 Early Heuristics:

One of the earliest methods proposed for SPLP is the now well known heuristic proposed by Kuehn and Hamburger [15]. It has been widely studied and its catalytic effect can be hardly overrated. The heuristic consists of two parts. The main program is an 'add routine' whereby facilities are located one by one corresponding to the greatest cost reduction until no facility can be added without increasing total cost. The underlying hypothesis is that the optimal solution for (p+1) facilities can be determined from the optimal solution for p facilities by adding an additional facility to the existing solution. In a modern technology, such a scheme would be called greedy because of its appetite for maximum improvement at each step. Upon termination of the main program, a so-called bump and shift routine is entered. It first eliminates (bumps) any facility which is now uneconomical because of the proximity of another facility located subsequently. It also considers relocating (shifting) each facility from its actual location to other potential locations in its neighborhood.

Manne [16] provided a Steepest Ascent One Point Move Algorithm (SAOPMA) for SPLP, a heuristic proposed be Reiter and Sherman for a more general class of combinatorial optimization problems. SAOPMA is a greedy improvement heuristic, initiated with a feasible solution in terms of y_i's which can be any of 2^m lattice points of the unit hypercube. It proceeds by moving to one of the m adjacent lattice points by replacing one of the y_i's with its complement, selected so as to give the greatest decrease in total cost thereby permitting both the addition of the new facility or the deletion of an existing facility. SAOPMA terminates with a suboptimal solution when movement to any adjacent lattice point will not result in a lower value of the objective function.

According to Feldman et al. [10], the SPLP –formulation is inadequate for those problems where the economies-of-scale affect facility costs over the entire range of facility sizes. In SPLP the concave objective function is a result of the fixed costs only; consequently a more general model is proposed:

$$\begin{aligned} \textbf{Min} \quad & \sum_i \sum_j c_{ij} + \sum_i f_i(s_i) \mid s_i = \sum_j s_{ij} \\ \textbf{s.t.} \quad & \sum_i s_{ij} = b_j, \forall j \\ & s_{ij} \geq 0, \forall i, j \end{aligned}$$

Where f_i (s_i) is continuous and concave over the range of interest.

The KH-heuristic (Kuehn and Hamburger) is based on sequential addition of facilities, assuming that the best p facilities in general will be a subset of the best p+1. They note however that facility might be eliminated (dropped) rather than added, i.e. facilities are in operation in every potential location in the first feasible solution and then eliminated one by one, guided by cost savings. Based on the single assignment property which is easily verified to apply for problems with any concave facility costs, Feldman et al. modified the KH-heuristic by incorporating a drop routine their computational results on the KH sample problems show that the running time for each instance is reduced to under one minute (IBM-7094) and that no solution obtained has a higher cost than that observed by Kuehn and Hemburger [15]. SAOPMA considered above consists of both add and drop procedures, where an odd or a drop can be performed at any step, in practice it usually performs like either drop or an add procedure, dependent only on the choice of initial solution.

Another effort on the heuristic frontier is that of Berghandahl. Since local minima can occur in nonconvex minimization, LP-techniques like separable programming can be expected to result in far-from-optimal local minima. Small-scale plant location problems due to Manne [16] are resolved by separable programming, confirming this expectation. A modification called 'Marginal Cost Parameterization' of the standard separable programming technique is therefore proposed leading to better solutions to the sample problems of a very moderate size (complete enumeration is faster), so, in spite of the acceptable solutions found, separable programming like techniques for SPLP and related problems cannot be recommended.

2.3 LP-relaxation of SPLP

LP-relaxation of SPLP have considered interest as they provide the basis both for various approximation algorithms for SPLP and for integer programming in general based upon branch and bound procedures. The following exposition characterizes such relaxations with respect to the quality of solutions and computational requirements.

For SPLP alternate ways of linking the fixed costs to facilities with positive outflow are either the strong (disaggregated) constraints or the weak (aggregated). Similarly, two alternate linear programming relaxations, SRS (strong relaxation of SPLP) and WRS (Weak Relaxation of SPLP) respectively, will be considered and are stated below.

$$\textbf{Min} \quad \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{i\in I} f_i y_i$$

$$\textbf{s.t.} \quad \sum_{i\in I} x_{ij} = 1, \qquad j \in J$$

$$x_{ij}\, \varepsilon\, 0, \qquad i \in I, j \in J$$

$$y_i = 0 \text{ or } 1, \qquad i \in I$$

Either $\quad y_i - x_{ij} \geq 0,\ i \in I, j \in J$ (SRS: strong relaxation of SPLP)

Or $\quad n_i y_i - \sum_{j\in J} x_{ij} \geq 0,\ i \in I$ (WRS: weak relaxation of SPLP)

Strong(SRS)	**Weak(WRS)**	**Both**
M inequalities	M inequalities	$\min \sum_i \sum_j c_{ij}x_{ij} + \sum_i f_i y_i$
$Yi - xi \geq 0$	$n_i y_i - \sum_i x_{ij} \geq 1$	$\sum_i x_{ij} \geq 1$
Disaggregated	Aggregated	$y_i \geq 0, x_{ij} \geq 0$

All slack variables for the tighter constraints y_i-x_{ij}≥0 are upwards bounded by +1 while slack variable corresponding to one of the aggregated inequalities in the more loosely constrained formulation can attain any non-negative value less than n_i, the number of clients which can be supplied from the i th facility. For n_i=n, each aggregated constraint is simply the sum of n disaggregated constraints. A feasible solution to SRS will therefore be feasible for WRS as well while the converse is generally not true. Thus, the adjectives 'strong' and 'weak' have the intuitive meaning that SRS with disaggregated constraints is stronger i.e. closer to the underlying SPLP than the weaker (less tight) from WRS. On the other hand, SRS has '*n(m+1)* constraints and '*m(2n+1)*' variables including slack while figures reduce to '*m+n*' and '*m(n+2)*' respectively for WRS.

2.4 Exact algorithms based on WRS or SRS

Credit for the first attempt to solve SPLP to optimally goes to Balinski and Wolfe [2] who devised an algorithm based on Benders decomposition. However, as reported the computational experience for that is discouraging. It therefore seems to attribute the first efficient algorithm for exact solution of SPLP to Efroymson and Ray [8].

The Effroymson-Ray algorithm, which was independently developed by Zimmerman, combines the then relatively novel branch and bound technique with WRS for providing lower bounds. Branchings are performed on the strategic y-variables with two branches emanating from each node corresponding to a selected, free y_i, fixed at either 0 or 1. Since WRS is solvable by inspection, several hundred bounds per minute could be evaluated on the computer IBM-7094. On the other hand, the number of positive x_{ij} found at each node is in general small compared to n_1; accordingly the amount of fixed cost absorbed by the i th facility is rather small, which in turn influences the tightness of the bounds obtained. Three simplifications have been incorporated to overcome this difficulty; simple tests are performed to check whether it pays to open or close certain facilities or to reduce some n_1's. The report on the computational efficiency is somewhat laconic. Efroymson and Ray [8] state that "a number of problems (m=50, n=200) have been solved in an average of 10 minutes on an IBM-7094". However, compared with earlier works, the results obtained are indeed significant for their time.

The adjective 'simple' in SPLP, now widely accepted as synonymous with 'uncapacitated' is originally coined by Spielberg. Some features of the algorithm presented resemble those of Efroymson and Ray [8] but their root is a general direct search mixed integer algorithm due to Lemke and Spielberg. For SPLP, the mixed integer algorithm operates on WRS and its dual which are solved for a sequence of fixed y-vectors, generated by a search algorithm due to Balas. This is a single branch scheme in contrast to Efroymson and Ray [8]. It starts with all y_I=0 and proceeds by assigning one's to selected y_I's on forward steps of the search. If the optimal solution has several open facilities, it can be expected that a good feasible solution will be found far away from the root of the search tree. It is then reasonable to relocate the search origin, say, to the best feasible solution known and to restart the search from that point.

The 'add' or 'drop' question which was briefly discussed in connection with heuristics plays an essential role in Spielberg's procedure. Two basic approaches with natural search origins – either all facilities open (drop) or all facilities closed (add) are investigated. If the fixed costs are relatively small or if the cost matrix is sparse (i.e. relatively few clients can be reached from each facility) it is reasonable to expect a large number of open facilities open (drop) to perform particularly well for such cases. Conversely, with large fixed costs and dense cost matrix, the natural search origin will be to close all facilities (add).

Spielberg sketched an "adaptive search algorithm" where the search origin is none of the extremes (all open, all closed) but preferably selected by the program itself, based on the actual data. The algorithms exploit the disaggregated form in terms of dual variable analysis leading to strong Bender's inequalities and gain functions. Both local and global gain functions are devised for curtailing the search. The local gains measure the local improvement of the objective function when a free facility is either opened or closed while the global gains at a certain node in the search tree measure the maximum improvement (over all successor nodes) incurred by a forward step. These conceptually simple devices together with different tests appear to be quite effective in looking ahead into the tree or to ascertaining optimality.

The mixed results obtained make it difficult to draw valid conclusions as to the data dependency of the algorithm's efficiency. Comparative results for programs with rigid origin exhibit in general a substantial decrease in the number of iterations required by the modified algorithms; however, the computing time per iteration increases considerably for these. Although approaches based on generalized origin seem promising for other problems of special structure and for integer programming in general, Spielberg summarizes his opinion of the SPL-situation with the following characterization: "An optimist might add that there is still ample room for improvements in strategy, corresponding to the great difficulties that still remain".

Two noteworthy exact WRS-based methods for SPLP were designed by Khumawallah and Hasen[. Since the later refers to the first, we start giving an account of the results achieved by Khumawallah whose approach can be regarded to be a significant improvement over the Efroymson-Ray algorithm with respect to storage requirements and computing time. Like his forerunner, Khumawallah operates on WRS but devices an efficient updating procedure, based on the three simplifications proposed by Efroymson and Ray. The purpose is to tighten the aggregated constraints $n_I\, y_I\text{-} \sum_j x_{ij}\quad 0$ by seeking n_I reduced to determine bounds at each node of the tree according to which certain facilities can be fixed either open or closed when the next branching is performed. Finally, an efficient storage scheme for bookkeeping information is developed.

Eight different branching decision rules are investigated and compared for 16 instances of SPLP (24, 50). The so-called 'Large Omega' rule, based on the maximum net gain achieved by opening a free facility appears to be superior in efficiency to the other branching rules examined. The average computing time with 'Large Omega' for the 16 problems is approximately 3.8 seconds on a CDC 6500. It is interesting to note, that Khumawallah's results lead to the following observation by one of his referees i.e. the problems for which there are about eight warehouses (facilities) open in the optimum solution seem to be most difficult; whereas, those with appreciably less than or greater than eight are relatively easier. Similar phenomena has been noticed by other researchers; the point is not that eight is a crucial number but that the optimum for these test data is flat in the sense that near-optimal solutions exist for a wide range of open facilities.

Hasen dealt with two implicit enumeration algorithms exploiting the concepts of additive penalties (akin to the gain functions introduced by Spielberg). Recalling the definition of the three disjoint index sets (S0, S1, SF) at each node of the search tree, a penalty can be viewed upon as increment of some lower bound z on the objective function z where a free variable y_1, is fixed at either 0 or 1, and if the addition of the corresponding penalties to a known lower bound can be shown to constitute a new valid bound, then these penalties are said to be additive.

2.5 Exact Dual Based Algorithms:

Here we simultaneously deal with two algorithms due to Bilde and Krarup[1] and Erlenkotter [9] which conceptually are closely related although their development spans about a full decade. Both methods, of which the latter is the most powerful, known to date for the exact solution of SPLP, the strong LP-relaxation of SPLP, and its dual DSRS.

DSRS:

$$\max z_{DSRS} \sum v_j \mid v_j - w_{ij} \le c_{ij}, \forall i, j$$

$$s.t. \sum_j w_{ij} \le f_i, \forall i$$

$$w_{ij} \ge 0, \forall i, j$$

and we observe that there exists an optimal solution to DSRS satisfying

$$\max\left(\sum_j v_j\right) = \sum \min\left(c_{ij} + w_{ij}\right) = z^0{}_{\mathrm{DSRS}} \tag{2}$$

Which leads to the following equivalent formulation in Bilde and Krarup [1] involving only the explicit variables w_{ij}.

Lower bound maximization problem

$$\max z_{DSRS} = \left(\sum \min(c_{ij} + w_{ij}) \middle| \sum w_{ij} \le f_i, \forall i\right) \tag{3}$$

$$\text{s.t.} \quad w_{ij} \ge 0, \forall i, j$$

since $z^0{}_{\mathrm{DSRS}} = z^0{}_{\mathrm{SRS}}$ by the strong duality theorem of LP, 'lower bound' above refers to the inequality $z^0{}_{\mathrm{DSRS}} \le z^0{}_{\mathrm{SPLP}}$. An alternate way of reasoning provides another equivalent form of DSRS, now involving only the v_j as explicit variables. For any set of given v_j's, we can without affecting the objective function assign the lowest possible value to all w_{ij}'s.

$$w_{ij} = \max\{0, v_j - c_{ij}\} \tag{4}$$

Substituting (4) into (2), the so called condensed dual is obtained.

Condensed Dual

$$\max z_{DSRS} = \left(\sum_j v_j \middle| \sum_j \max\{0, v_j - c_{ij}\} \leq f_i, \forall i \right) \tag{5}$$

Evidently, a lower bound on $z^0{}_{\text{SPLP}}$ can be found by solving DSRS or one of its two equivalent forms. But since the idea of the two dual based algorithms is to combine LP with branch and bound and since lower bounds normally have to be generated repeatedly throughout such computations, we seek a bounding procedure which rather than striving after an optimal solution to DSRS exploits its structure in an attempt to compromise between sharp bounds and limited computational effort.

In the early wok done by Bilde and Krarup [1], a conceptually straight forward heuristic is developed for finding near optimal solutions and later on incorporated as the basic element in a series of branch and bound algorithms by Bilde and Krarup [1]. In addition, the method has proven to be quite powerful for solving problems of moderate size by hand (optimal solutions to real-world problems with up to 29 potential facilities are found and verified with in a few minutes). However, the accessibility of the Bilde-Krarup approach and the computational results published up to 1970 were limited due to their appearance in Danish; a version with minor revisions is now available in English as Bilde and Krarup [1]. With such an impractical publication in mind, it is reasonable to claim that the BK-approach has been independently discovered by Erlenkotter [9] as a central part of his DUALOC-algorithm. While the two methods, the lower bound maximization heuristic and Erlenkotter's Dual Ascent procedure operating on the condensed dual, essentially are identical (both will be referred to as dual ascent in the sequel), Erlenkotter [8] has added further refinements which makes DUALOC superior to any other exact algorithm proposed.

Throughout the computations $v_j = \min\{c_{ij} + w_{ij}$. A cell (i, j) is called admissible, if $v_j = c_{ij}+w_{ij}$ and we denote by A_j the index set of admissible cells in column j. Dual ascent is initiated with all $w_{ij} = 0$, hence the initial value of v_j is min c_{ij} for all i. In each iteration we select a column k and increase some of the w_{ik}'s by an amount $\Delta > 0$ so as to increase $\min\{c_{ij} + w_{ij}\} = v_k$ for all i. To obtain the desired effect we must simultaneously increase (at least) all w_{ik}'s corresponding to the admissible cells. Unless $A_k = I$, at least one new admissible cell will appear in column k provided that none of the constraints $\sum_j w_{ij} \le f_i$ are violated. Let I^+ be the index set of rows for which $\sum_j w_{ij} \le f_i$ hold as equalities, i.e. $I^+ = \left\{ i \middle| \sum_j w_{ij} = f_i \right\}$. Clearly, a column k for which $I^+ \cap A_k \neq 0$ is no longer a candidate for further ascent. Thus the process terminates when $I^+ \cap A_k \neq 0$ for all j.

Upon the termination of dual ascent, a feasible integer solution to SRS, and to SPLP is readily available. Since each column has at least one admissible cell in a row belonging to I^+ (the halt criterion), such a solution can be generated as follows:

For each j, let $i^+(j)$ be the index set of rows defined by $i^+(j) = \{i | c^+_{i(j),j} = \min_{i \in i^+} \{c_{ij}\}\}$; Clearly, all $c^+_{i(j),j}$ correspond to admissible cells. A feasible integer solution (y^+, x^+) is then determined by

$$y^+ = 1 \text{ for } i \in i^+$$

$$= 0 \text{ otherwise} \tag{6}$$

$$x^+_{ij} = 1 \text{ for some } i \in i^+(j), \text{ for all } j,$$

$$= 0 \text{ otherwise} \tag{7}$$

where ties in case of several $i \in i^{+}(j)$ are resolved arbitrarily.

For all i, j let w_{ij}^{+} and v_i^{+} =min$_i\{c_{ij} + w_{ij}\}$ be final values of w_{ij}, v_j found by dual ascent. (y^{+}, x^{+}) is feasible for SRS and (w^{+}, v^{+}) is feasible for DSRS; these solutions are thus optimal for SRS and DSRS respectively if they satisfy the complementary slackness conditions.

$$\left(c_{ij} + w_{ij}^{+} - v_j^{+}\right)x_{ij}^{+} = 0, \forall i, j \tag{8}$$

$$\left(f_i - \sum_j w_{ij}^{+}\right)y_i^{+} = 0, \forall i \tag{9}$$

$$\left(y_i^{+} - x_{ij}^{+}\right)w_{ij}^{+} = 0, \forall i, j \tag{10}$$

c_{ij}+ w_{ij}^{+} − v_i^{+} = 0 indicates that $(i, j$ is admissible. $(f_i - \sum_j w_{ij}^{+}) = 0$ implies that $i \in \mathrm{I}^{+}$.

Hence equation (8) and equation (9) are automatically satisfied by the solution (y^{+}, x^{+}) provided by equation (7). If, in addition equation (10) is satisfied, then (y^{+}, x^{+}) is optimal not only to SRS but also to the underlying SPLP.

While Bilde and Krarup [1] made rather vague observations as to whether or not an optimal solution to SPLP is readily available upon termination of dual ascent, Erlenkotter [9] devised a dual adjustment procedure to be used unless above equations are satisfied. The idea of dual adjustment is to select some column k for which equation (10) is violated. Such a column must contain at least two admissible cells since all positive w_{ik}^{+} appear in admissible cells only and since exactly one x_{ik}^{+} is equal to one. If we simultaneously decrease all w_{ik}^{+}, $i \in \mathrm{A_k}$, by Δ, then v_j^{+} will be decreased by Δ and the previously binding constraints $\sum_j w_{ij} = f_i, \mathrm{i} \in \mathrm{I}^{+} \cap \mathrm{A_k}$, will be replaced by $\Delta + \sum_j w_{ij} = f_i$. It is then attempted to increase other w_{ij} in order to increase other v_j whereby the dual objective function may increase and a new integer solution (y^+, x^+) may result. Dual adjustment can thus viewed as a 'relocation of the resources available' in an attempt to close the duality gap by reducing the complementary slackness violations.

If the dual ascent and adjustment procedure fails to close the gap, a branch and bound phase follows, that uses the bounds provided by these procedures. Since the bounds have proven to be so effective that more sophisticated versions seem unnecessary, Erlenkotter [9] simply branches on the lowest cost facility $i^+(j)$ contributing to the violation of a complementary slackness condition.

2.6 Dual Ascent and Lagrangian Relaxation

A general theory of Lagrangian relaxation which has provided a unifying framework for several bounding procedures in discrete optimization has been developed in Geoffiron with particular emphasis on LP-based branch and bound. Since the most powerful techniques known for solving SPLP, hitherto culminating with Erlenkotter's DUALOC, can be viewed upon as parameterized Lagrangian relaxation.

For a general integer LP problem

$$\min z = \{cx | Ax \geq b\}, Bx \geq d, x \geq 0, \text{some } x_{\mathrm{I}} \text{ integer} \quad (11)$$

The Lagrangian relaxation above relative to the constraint set $Ax \geq$ b and a conformable nonnegative vector λ is defined by

$$\min z_L = \{cx + \lambda(b - Ax) | Bx \geq d\}, x \geq 0, \text{some } x_{\mathrm{I}} \text{ integer} \quad (12)$$

The essence of Lagrangian relaxation is to identify a set of 'complicating constraints' ($Ax \geq$ b), weight these by multipliers and insert them in the objective function in order to obtain a new problem which, hopefully, is simpler to solve than the original. One of the earliest and perhaps most convincing examples of this technique is the work of Held and Karp on the symmetric traveling salesman problem of finding a minimum-weight 1-tree, which is solvable in polynomial time by the greedy algorithm. Lagrangian relaxation can thus be viewed as an alternative to LP and frequently a very attractive alternative – as a means of providing bounds in a branch and bound algorithm.

Two different Lagrangian relaxations of SPLP almost suggest themselves. Within the context of the dual-based algorithms discussed above, the first is obtained by taking yi-$x_{ij}\geq 0$ the set of 'complicating constraints' with the w_{ij}'s as the corresponding set of nonnegative multipliers.

Lagrangian relaxation of SPLP

$$\min z_L(w) = \sum_i \sum_j (c_{ij} + w_{ij})x_{ij} + \sum_i \left(f_i - \sum_j w_{ij} \right) y_i \tag{13}$$

$$\sum_i x_{ij} = 1, \forall j$$

$$x_{ij} \geq 0, \forall i, j$$

$$x_{ij} \geq 0, \forall i, j$$

The second Lagrangian relaxation is obtained by taking $\sum_i x_{ij} = 1, \forall j$, as the set of 'complicating constraints'. If the values of the multipliers are fixed, the y-variables can be determined by inspection. Then above can be written as

$$\min z_L(w) = \sum_i \sum_j (c_{ij} + w_{ij})x_{ij} + \sum_i \min\left\{ f_i - \sum_j w_{ij}, 0 \right\} \left| \sum_i x_{ij} = 1 \right.$$

s.t. $\quad x_{ij} = 0, \forall\ i,j$

If for some i$\{\min(f_i - \sum_j w_{ij}, 0)\}$is negative, then its value is can be increased to 0 by decreasing $\sum_j w_{ij}$ until this sum equals f_i. Such a change in the w_{ij} will increase the minimum value of the objective function. Therefore $z_L(w)$, only nonnegative w_{ij} satisfying $\sum_j w_{ij} \leq f_i$ should be considered. Hence, with the formulation of DSRS in mind, we can appropriately define the set of feasible multipliers w_{ij} to assure dual feasibility by $\left\{ w_{ij} \middle| \sum_j w_{ij} \leq f, \forall i, w_{ij} \geq 0, \forall i, j \right\}$. For such w, $z_L(w)$ is solvable by inspection; the minimum value $z_L^0(w)$ of $z_L(w)$ is simply the sum of the column minima of the $(C+W)$-matrix.

$$z_L^0(w) = \sum_j \min\{c_{ij} + w_{ij}\} \forall i. \tag{14}$$

equation (14) is recognized as the expression used for previously for removing the v_j variables from DSRS on our way to the simplified formulation (4). Equation (14) in other words, the lower bound maximization problem for fixed multipliers w_{ij}, but the quality of the bound thus obtained is, of course, strongly dependent on how the multipliers are selected.

The sharpest bound is found by maximizing $z_L^0(w)$ over the set of feasible multipliers

$$\max{}_w z_L^0(w) = \left\{ \sum_j \min\{c_{ij} + w_{ij}\} \middle| \sum_j w_{ij} \leq f, \forall i_i \right\} \tag{15}$$

s.t. $w_{ij} \geq 0, \forall\ i,j$

This is seen to coincide with the bound, obtained via the LP-relaxation (4) or its equivalent form (6). Equation (15) is also known as the formal Lagrangian dual of equation (11) with respect to the constraints $Ax \geq b$ since its solution represents the best choice of multipliers. We note that this result whereby both Lagrangian relaxation and DSRS lead to the lower maximization problem for determining bounds on the optimal solutions of an SPLP does not hold in general for an arbitrary combinatorial optimization problem. In fact for a given integer programming formulation it can be shown that the Lagrangian relaxation will provide at least as sharp bounds if the bound provided by the Lagrangian relaxation is not increased by relaxing the integrality restrictions in which case the Lagrangian relaxation is said to possess the Integrality property. Thus to summarize: in terms of Lagrangian relaxation where the dual ascent (for DUALOC supplemented by dual adjustment) is an approximate algorithm for setting the parameters (multipliers) in an attempt to obtain the best possible bounds.

2.7 Dual Based Procedure for Uncapacitated Facility Location by Erlenkotter [9]

Model formulation and Solution procedure

To formulate a model for the uncapacitated facility location problem, Erlenkotter [9] defined the following notation

x_{ij} = Fraction of location, $j \in J$'s demand supplied from facility

y_i = 1, if facility I is established

= 0, otherwise

c_{ij} = total of variable capacity, production, and distribution costs for supplying all of location j's demand from facility i

$f_i \geq 0$ if fixed cost for establishing facility i.

$$\textbf{Min} \quad z_P = \sum_{i=I}\sum_{j=J} c_{ij}x_{ij} + \sum_{i=I} f_i y_i \tag{16}$$

$$\textbf{s.t.} \quad \sum_{i=I} x_{ij} = 1, j \in J \tag{17}$$

$$y_i - x_{ij} \geq 0, i \in I, j \in J \tag{18}$$

$$x_{ij} \geq 0, i \in I, j \in J \tag{19}$$

$$y_i \in \{0,1\} i \in I \tag{20}$$

in an attempt to obtain a natural solution, we will solve the linear programming relaxation that replaces the equation (20) by

$$y_i \geq 0,\ i \in I. \tag{21}$$

The dual problem for (16)-(19) and (21) is

$$\max\ z_D = \sum_{j=J} v_j \tag{22}$$

$$\text{s.t.} \quad \sum_{j=J} w_{ij} \leq f_i, i \in I \tag{23}$$

$$v_j\text{-}w_{ij} \leq c_{ij},\ i \in I,\ j \in J \tag{24}$$

$$w_{ij} \geq 0,\ \mathrm{i} \in \mathrm{I},\ \mathrm{j} \in \mathrm{J} \tag{25}$$

For any feasible choice of the dual variables v_j, setting each variable w_{ij} at the lowest value possible will maintain feasibility and leave the value of the objective function unchanged. Thus we assume

$$w_{ij} = \max\{0,\ v_j\text{-}c_{ij}\} \tag{26}$$

substituting equation (26) into (23) replaces (22)-(25) with the following condensed dual that involves only the explicit variables v_j.

A violation of (30) occurs whenever, for some j, more than one $\mathrm{i} \in \mathrm{I}^+$ has $c_{ij} < v_j$

since $x_{ij}^+ = y_i^+ = 1$ for the lowest value of c_{ij} only.

Linear programming theory provides a simple relationship between these complementary slackness violations and the difference between the dual objective value z_D^+ for $\{ v_j^+ \}$ and the primal objective value z_P^+ corresponding to the integer solution (32)-(33). We start this result as a lemma and show it directly.

LEMMA.

$$z_P^+ - z_D^+ = \sum_{j \in J} \sum\nolimits_{i \in I^+, i \neq i^+(j)} \max\{0, v_j^+ - c_j\}.$$

PROOF

$$\begin{aligned} z_D^+ &= z_D^+ + \sum_{i \in I^+} \left[f_i + \sum\nolimits_{j \in J} \max\{0, v_j^+ - c_{ij}\} \right] \\ &= \sum\nolimits_{j \in J} v_j^+ + \sum\nolimits_{i \in I^+} f_i + \sum\nolimits_{j \in J} \left(c_j^+ - v_j^+ \right) - \sum\nolimits_{j \in J} \sum\nolimits_{i \in I^+, i \neq i^+(j)} \max\{0, v_j^+ - c_{ij}\} \\ &= z_P^+ - \sum\nolimits_{j \in J} \sum\nolimits_{i \in I^+, i \neq i^+(j)} \max\{0, v_j^+ - c_{ij}\}. \end{aligned}$$

Clearly, an integer primal solution (32)-(33) that exhibits no complementary slackness violations is optimal. Our solution approach will attempt to close the gap between primal and dual solutions by reducing these violations.

A set Γ^+ may exclude some facility locations I in the eligible set $I^* = \{i : \sum\nolimits_{j \in J} \max\{0, v_j^+ - c_{ij}\} = f_i\}$ we require only for each j that $v_j^+ \geq c_{ij}$ for some $i \in \Gamma^+$.

Adding an additional facility i^1 to a minimal set Γ^+ that satisfies this condition cannot improve the solution i^1 to a minimal set Γ^+ that satisfies this condition cannot improve the solution. From the Lemma the inclusion of i^1 changes the primal objective value z_P by $\sum \max\{0, v_j^+ - \max\{c_j^+, c_{i^1 j}\}\} \geq 0$. conversely, deletion from a set Γ^+ of a non-essential facility i^1 that is not required for such a minimal set cannot make the solution worse.

If more than one facility can be designated as non-essential, the construction of a best minimal set Γ^+ is a combinatorial problem itself. Since the solution approach does not require a best minimal set, we use a rapidly obtained approximation to one. Firstly, we include in Γ^+ all essential facilities, i.e., any $i \in \Gamma^*$ for which only a single $c_{ij} \leq v_j^+$ for $i \in \Gamma^*$ and some j. we then examine sequentially all j for which there is no $i \in \Gamma^+$ with $c_{ij} \leq v_j^+$ and augment Γ^+ by the eligible facility $i \in \Gamma^*$ that has minimum c_{ij}.

2.8 Dual Solution Procedure

The dual problem has a very simple structure and typically has multiple solutions (primal solutions are usually quite degenerate). Since an exact solution to the relaxed linear problem does not always yield an optimal integer primal solution, we should not be obsessed with obtaining an exact solution. Resolution of these non-integer solutions seems best accomplished through a branch and bound phase. A simple approach that provides good bounds from sub-optimal dual solutions will be adequate.

The first component of our solution approach is a dual ascent procedure that constructs a dual solution $\{ v_j^+ \}$ and an associated set of facility locations Γ^+, with the properties described earlier. This procedure begins with any dual feasible solution $\{v_j\}$ and repeatedly cycles through the demand locations 'j' one by one attempting to increase v_j to the next higher value of c_{ij}. This incremental approach to increase v_j appears to reduce the likelihood of complimentary slackness violations by initials disturbing the number of $c_{ij} \leq v_j$ evenly among the demand locations j. If constraint (28) blocks the increase of v_j to the higher c_{ij}, v_j is increased to the maximal level allowed by the constraint. When all the v_j are blocked from further increases, the procedure terminates.

We will specify the procedure formally. For notational and computational convenience, where index c_{ij} for each j in non-decreasing order as c_j^k, $k = 1, \ldots.., m$, and include a high-cost dummy source with c_j^{m+1} =+∞... To obtain an initial feasible dual solution, set $v_j = c_j^1$ for each j. For use in subsequent adjustment procedure, we restrict changes in the v_j to a subset of demand locations $J^+ \subseteq$ J. If all v_j are eligible for change, as in the initial application of the procedure, we set $J^+ =$ J. We now give the dual ascent procedure.

2.8.1 *Dual Ascent Procedure*

1. Initializing with any feasible dual solution $\{v_j\}$ such that

$$v_j \geq c_j^1 \text{ for } j \in \mathrm{J}$$

$$s_\mathrm{I} = f_i - \sum_{j \in J} \max\{0, v_j - c_{ij}\} \geq 0, \forall i \in I.$$

for each $j \in$ J, defining $k(j) = \min\{k{:}v_j \leq c_j^k\}$. If $v_j = c_j^{k(j)}$, increase $k(j)$ by 1.

2. Initializing j =1 and $\delta = 0$.

3. If $j \in J^+$,go to step 7.

4. Set $\Delta_j = \min_{i \in I}\{s_i : v_j - c_{ij} \geq 0\}$.

5. If $\Delta_j > c_j^{k(j)} - v_j$, set $\Delta_j = c_j^{k(j)} - v_j$ and δ =1 and increase $k(j)$ by 1.

6. Decrease s_i by Δ_j for each $i \in$I with $v_j - c_{ij} \geq 0$; then increase v_j by Δ_j.

7. If $j \neq n$, increase j by 1 and return to step 3.

8. If $\delta = 1$, return to Step 2. Otherwise, terminate.

Since each v_j is increased until blocked by an equality in constraint (28), the final solution from this procedure with $J^+ =$ J satisfies the condition required for a solution $\{v_j^+\}$ and facility set $I^{+} \subseteq I^{*} = \{i : s_i = 0\}$. Non-essential facilities can be excluded from I^{+} to improve the corresponding primal solution (32)-(33).

As described, the dual ascent procedure arbitrarily processes demand locations according to ascending order. We have experienced with two other processing strategies, decreasing order and alteration between ascending and descending order at each passage through step 2 of the procedure. Different strategies can lead to somewhat different solutions, and none has been uniformly superior. The alternating strategy has been the most consistent and was used for all the computational results reported here.

The dual ascent procedure yields a candidate dual solution and a candidate integer primal solution through (32)-(33). If these solutions satisfy all the complementary slackness conditions (30), the solutions are optimal. If not, we try to improve the solutions through a dual adjustment procedure. To initiate this adjustment procedure, we select some j^1 for which (30) is violated. Suppose we decrease v_j. This creates slack on at least two binding constraints (28), corresponding to those $i \in I^{+}$ with $v_{j1} > c_{ij1}$. We then attempt to increase other v_j that are limited by these constraints. If more than one v_j can be increased unit for unit as v_j, is decreased, the dual objective value will be increased by this change. If only one v_j can be increased, the dual objective value will remain at the same level, but the slack created on constraint (28) will alter the primal solution since I^{+} will be changed. We cycle through this process for all locations j^1 and may repeat the procedure as long as the dual objective continues to improve. Although more complex adjustment rules can be devised, we have not implemented them since this simple procedure has been very effective.

These changes in the dual solution are accomplished conveniently with the dual ascent procedure. We decrease v_j identified as likely candidates for increase. A final pass of the dual ascent procedure with $J^+ = J$ completes the adjusted solution.

To specify this procedure formally, defining the following additional notation:

$$I_j^* = \{i \in I^* : v_j \geq c_{ij}\} \text{ for all } j \in J$$

$$I_j^+ = \{i \in I^+ : v_j > c_{ij}\} \text{ for all } j \in J$$

$$J_i^+ = \{j : I_j^* = \{i\}\} \text{ for all } i \in I$$

Supplementing the definition of $i^+(j)$ in last , we define a second- best source $i^1(j) \in I^+$ with

$$c_{i^1(j),j} = \min_{i \in I^+, i \neq i^+(j)} c_{ij}, \text{ for all } j \in J \text{ with } |I_j^+| > 1,$$

$$c_j^- = \min_{i \in I} \{c_{ij} : v_j > c_{ij}\} \text{ for } j \in J$$

If $|I_j^+| > 1$, we have a violation of the complementary slackness condition (30); if $|I_j^+| > 1$, for each $j \in J$, the solution (32) corresponding to I^+ is optimal. If $|I_j^*| = 1$, a single constraint (28) for $i^+(j)$ blocks v_j from increasing, and therefore v_j is candidate for increase if some v_j contributing to the left-hand side of this constraint is decreased.

2.8.2 *Dual Adjustment Procedure*

1. Initialize $j = 1$.

2. If $|I_j^+| > 1$, go to step 7.

3. If $J_{i^+(j)}^+ = \Phi$ and $J_{i^1(j)}^+ = \Phi$, go to step 7.

4. For each $i \in I$ wit $v_j > c_{ij}$, increase s_i by $v_j - c_j^-$, then decrease v_j to c_j^-.

5. (a) Set $J^+ = J_{i+(j)}^+ \cup j_{i(j)}^+$ and execute the dual ascent procedure.

(b) Augment $J^+$by J and repeat the dual ascent procedure.

(c) Set J^+=J and repeat the dual ascent procedure.

6. If v_j has not resumed its original value, return to step 2.

7. If j 1 n, increase j by 1 and return to step 2. Otherwise, terminate.

Since we are making a discrete reduction in v_j, the purpose of step 5(b) is to increase v_j to take up any of this decrease that has not been absorbed by increase in the other dual variables. Each unit of decrease for v_j in the final solution is matched by a unit increased in at least one other dual variable, and the value of the dual objective therefore cannot be decreased by this procedure. We execute the dual ascent procedure in step 5(c) with J^+=J to ensure termination with a valid solution $\{v_j^+\}$. If the dual adjustment procedure increases the value of the dual objective, repetition may give further improvement.

If the dual ascent and adjustment procedures do not yield an optimal integer solution to (1) - (5), we complete the search with a branch-and-bound phase that uses as bounds the solutions provided by these procedures. The branch-and-bound approach is the simplest possible, the bounds have been so effective that a more elaborate version seems unnecessary. Salient features of this branch and bound phase are

a) For branching, we select some facility location contributing to the violation of complementary slackness condition (30). No attempt is made to discriminate among possible choices- ate the first violation encountered, we branch on the corresponding lowest-cost source $i^+(j)$.

b) For ease in updating the solution and restarting when backtracking, initially we always fix the branching facility closed.

c) As described by Geoffiron, an elementary backtracking scheme with last-in, first-out processing of nodes minimizes computer storage requirements and simplifies updating of solutions.

d) To fix facility i closed, we replace the fixed charge f_i by $+\infty$. The current dual solution remains feasible. Application of the dual ascent and adjustment procedures may improve the bound given by the value of the dual objective through increasing those v_j no longer restricted by the dual constraint(28) for i.

e) To fix facility i open, were place the fixed charge f_i by 0 and restore dual feasibility by reducing the value of each $v_j > c_{ij}$ to c_{ij}. Adding to (13) the fixed charges f_i for all facilities fixed open gives the value of the dual objective for the restricted problem. Since $\sum_{j=J} \max\{0, v_j - c_{ij}\} = f_i$ for $i \in I^+$, reduction of those $v_j > c_{ij}$ does not change the dual objective value. Reducing these v_j is likely to add slack to other dual constraint (14), and apply the dual ascent and adjustment procedures may increase the value of the dual objective.

f) Fathoming of nodes in the branch and bound phase is either by bounding or by obtaining a primal integer solution (32)-(33) that also satisfies the complementary slackness conditions (30).

g) At each node we construct primal integer solutions (32)-(33) only after the initial application of the dual ascent procedure and after each completion of the dual adjustment procedure, and not each time the ascent procedure is completed with $J^+ = J$.

After investigating four levels of dual improvement efforts, all levels apply the dual ascent procedure at each node. The first level (no dual improvement) excludes the dual adjustment phase entirely. The second level (one pass dual improvement) applies the dual adjustment procedure once at each node with repetition only when the primal solution improves. The fourth level (maximum dual improvement) repeats the dual adjustment procedure at each node as long as the dual objective value continues to increase. The third level (maximum/one pass dual improvement) is a compromise between the second and the fourth levels that obtains maximum dual improvement at the initial node and one pass improvement at subsequent nodes. The rationale for this level is that the tightest possible bound for the base solution at the initial node is valuable as a starting solution, but exclusive effort at subsequent nodes may yield only marginal improvement. Higher levels of dual improvement typically lead to less branching. Although further dual improvement might be possible through sub-gradient optimization, this option was not explored since the third and fourth levels of improvement have provided optimal linear programming solutions to most test problems without branching.

2.9 New Formulation and Relaxation of SPLP given by Sharma and Muralidhar

2.9.1.1 *Variable Definition*

Quantity received by market '*k*' from plant '*i*' as a fraction of total market demand is represented by X_{ik}. If decision is to locate a plant at point '*i*' then $Y_i = 1$ and equal to zero otherwise.

2.9.1.2 Constants *of the problem*

D_k is the demand at market '*k*' and d_k is the demand at market '*k*' as a fraction of total market demand. Then

$$d_k = \frac{D_k}{\sum_{k=1}^{K} D_k}$$

Total numbers of markets are *K*. Total number of possible plant locations are *I*. If decision is to locate a plant at potential location '*i*' then we incur a fixed cost of f_i. Plants are assumed to have unlimited capacities. Cost of transporting $\sum_{k=1}^{K} D_k$ units of goods from plant '*i*' to market '*k*' is denoted by C_{ik} .This means C_{ik} is the cost of transporting total market demand (of all markets) from plant *i* to market *k*. This new definition has been used by Sharma and Sharma [19] where they give a new formulation of the transportation problem. New formulation of SPLP was developed by using that.

2.9.1.3 *Formulation of Simple Plant location Problem.*

Problem SPLP: min $\sum_{i,k} X_{ik} C_{ik} + \sum_{i} Y_i . f_i$

$$s.t. \quad \sum_{i}\sum_{k} X_{ik} = 1 \quad (1)$$

$$-\sum_{i} X_{ik} \geq -d_k, \forall k \quad (2)$$

$$Y_i \geq \sum_{k} X_{ik}, \forall\, i = 1..I \quad (3)$$

$$X_{ik} \geq 0, \forall i, k \quad (4)$$

$$Y_i = (0,1), \forall i \quad (5)$$

Equation (1) ensures that adequate quantities are shipped from plants to warehouses so that total demand can be met. Equations (1) and (2) ensure that at any market we supply quantities, which is exactly equal to the demand at that market. Equations (3) ensures that if a plant is not located at any plant location then quantities shipped out of that location is equal to zero. Equations (4) is the non-negativity constraint and equations (5) are integer restrictions of variable Y_i. In the next section we describe the solution procedure to solve the linear programming relaxation of problem SPLP.

2.9.2 Solving the Linear Programming Relaxation of SPLP

We relax the integer restrictions (5) to obtain the linear programming relaxation of problem SPLP which is given below.

Problem RP: $\quad \min \quad \sum_i \sum_k X_{ik} C_{ik} + \sum_i f_i y_i$

$\quad$ s.t (1) to (4) and $Y_i \geq 0, \forall i$

We associate v_o, v_k and w_i as dual variables respectively with equations (1), (2) and (3) to obtain dual of problem RP as given below.

Problem DRP: $\quad \max \quad v_0 - \sum_k d_k v_k$

$$\text{s.t.} \quad v_0 - v_k - w_i \leq C_{ik} \tag{6}$$

$$w_i \leq f_i, \forall i \tag{7}$$

$$v_k \geq 0, \forall k \tag{8}$$

$$w_i \geq 0, \forall i \tag{9}$$

$$v_0 \text{ unrestricted in sign} \tag{10}$$

Now in theorem 1 below, we prove a simple result that will later help us in obtaining the optimal solution to problem to DRP.

Theorem 1. There exists an optimal solution to problem DRP with $w_i = f_i$ (assuming $f_i \geq 0$, *for all i*)

Proof: If the optimal solution to the problem DRP has all $w_i = f_i$, then we have nothing to prove. Assume we have an optimal solution to problem DRP with some $w_i < f_i$ and then we obtain a new solution to problem DRP with the same objective function value and all the constraints still being satisfied (as $f_i \geq 0$ for all i) by setting those $w_i = f_i$.

Now we give algorithm A1 below to solve the problem DRP.

2.9.2.1 Algorithm for solving the problem DRP:

Algorithm A1:

Step 1: Using the results of theorem 1, we set $w_i = f_i$, *for* $i= 1..I$ and the problem DRP reduces to the following.

Problem DRP1: max $v_0 - \sum_k d_k v_k$

$$\text{s.t.} \quad v_0 - v_k \leq C_{ik} + f_i, \quad \forall i \text{ and } k \tag{11}$$

Step 2: Now find $d^*_k = \min_i \, (C_{ik} + f_i)$ *for all* $k=1..K$ and remove all the redundant constraints (of equations (11)) in problem DRP1. In the case of a tie, that is $C_{ik} + f_i = d^*_k = C_{i_1 k} + f_{i_1}$ only one equation is retained and all others are eliminated. Then the problem DRP1 becomes the following.

Problem DRP 2: max v_0 - $\sum_k d_k v_k$

$$\text{s.t} \quad v_0 - v_k \leq d_k^* \;\; \textit{for all } k=1..K \tag{12}$$

Step 3: We sort the values of d^*_k in an increasing order and re index such that

$$\mathrm{d}_k^* \leq d_{r+1}^* \qquad \textit{for all } r = 1..K\text{-}1.$$

Step 4: Since $\sum_{k=1}^{K} d_k = 1$, then

$$\text{we let} \quad v_0 = d_K^*, v_k = v_0 - d_k^* \qquad \textit{for all } k.$$

Now we have the following theorem :

Theorem 2: If primal problem RP is feasible, then algorithm A1 produces an optimal solution to problem DRP.

Since $\sum_{k=1}^{K} d_k = 1$, we let $v_0 = d_K^*, v_k = v_0 - d_k^*$ *for all k*; then any increase in value of v_0 leads to no change in objective function value of problem DRP. Now we have the following simple result.

Result 1: Optimal solution to problem DRP is $\sum_k$ ***$d_k(\min_i(C_{ik} + f_i))$.***

Proof: It is easy to see.

Now we give a procedure, which prepares the optimal solution to problem RP.

2.9.2.2 *Constructing an optimal solution to problem RP*

Algorithm A2:

Step1: We set the dual variables X_{ik} = 0, associated with the redundant constraints identified in step 2 (eqn 11) of algorithm A1.

Step 2: The remaining dual variables X_{ik} associated with binding dual constraints are set as follows:

Since $\sum_{k=1}^{K} d_k = 1$, then we set

$X_{i1,k} = d_k$, for some $i = i^1$and $k : d_k^* = C_{i1k} + f_{i1}$ and

X_{ik}= 0 for all i: $(i \neq i_1)$ and $k\, d^*_k \leq C_{ik} + f_i$

Step 3: Finally set $Y_i = \sum_{k=1}^{K} X_{ik}$ *for all i=1..I.*

It may be noted that in step 2 when d^*_k is not unique for some k, the algorithm sets X_{ik}=d_k for exactly one i for which d^*_k= C_{ik}+f_i. We show that the algorithm A2 produces optimal solution to problem RP in the theorem 3 below.

Theorem 3: Algorithm A2 produces the optimal solution to the problem RP.

Proof: It can be easily verified that the dual solution prepared by algorithm A1 and the associated primal solution given by algorithm A2 satisfy the complementary slackness conditions.

Result 2: the optimal solution to problem RP1 is obtained in O (n^2) number of steps (where n is the sum of I and K).

Proof: Step 2 of algorithm A1 takes O (IK) number of steps as we find minimum among I values K number of times. Step 3 of algorithm A1 take O (K log K) number of steps. Hence complexity is dominated by the Step 2 of algorithm A1. If 'n' is the sum of warehouse points and the supply points, then the optimal solution to problem RP is obtained in O (n^2) number of steps.

2.9.3 Relative strengths of different relaxations of SPLP

In literature it has been already established that the bounds given by the strong relaxation of SPLP (SRS) are better than the bounds given by the weak relaxation of SPLP (WRS), see Bilde and Krarup [1]. Here relative strengths of SRS and WRS (new formulation) are discussed.

2.9.3.1 Comparing bounds given by SRS and the new relaxation developed

We cast the "strong relaxation" of SPLP using the decision variables and the structure of formulation given in section two. Thus SRS formulation is as given below:

Problem SP:
$$\min \sum_i \sum_k X_{ik} C_{ik} + \sum_i f_i Y_i$$

$$\text{s.t.} \quad \sum_i \sum_k X_{ik} = 1 \qquad (13)$$

$$- \sum_i X_{ik} \geq - d_k \; \forall k \qquad (14)$$

$$d_k Y_i - X_{ik} \geq 0 \, \forall i,k \qquad (15)$$

$$X_{ik} \geq 0, \quad \forall i,k \qquad (16)$$

$$Y_i = (0,1), \quad \forall i, \tag{17}$$

Now the strong relaxation of SPLP in the newer formulation is

Problem RSP: min $\sum_i \sum_k X_{ik}C_{ik} + \sum_i f_i Y_i$

s.t. (13)-(16) and $Y_i \geq 0, \ \forall i.$

We now associate dual variables v_0 with equation (13), v_k with equations (14), and w_{ik} with equations (15) and write the dual of problem RSP as follows:

Problem DRSP: max $v_0 - \sum_k d_k v_k$

$$s.t. \quad v_0 - v_k - w_{ik} \leq C_{ik} \tag{18}$$

$$\Sigma_i \, d_k \, w_{ik} \leq f_i \tag{19}$$

$v_k \geq 0, \forall$ k,; $w_{ik} \geq 0, \forall$ i, k; v_0 unrestricted in sign.

In the problem DRSP if we set $w_{ik} = f_i \ \forall k, \forall i$, then equations (19) are satisfied as equality constraints and resulting problem is similar to problem DRP1 and hence it follows that then the problem DRSP has $\sum_k$ $d_k(min_i(C_{ik} + f_i))$ as its objective function value.

Now this is same as the bound given by the new relaxation presented in section 3 (see result 1). However with these dual values, we are unable to satisfy complementary slackness conditions for equations (16), and hence the current dual solution is not optimal to problem DRSP. Hence we have the following result.

Result 3: The bounds given by new relaxation for SPLP in section three are worse than the bounds given by SRS.

Proof: It is easy to see from above.

Now we determine relative strength of WRS and the new relaxation developed in this paper.

2.9.3.2 Comparing the bounds given by WRS and the new relaxation developed

Now we cast the "Weak Relaxation" of SPLP using our decision variables and the structure of formulation given in section two. Thus WRS formulation is as given below:

Problem WP: min $\sum_i \sum_k X_{ik}C_{ik} + \sum_i f_i Y_i$

$$\text{s.t.} \quad \sum_i \sum_k X_{ik} = 1 \tag{21}$$

$$- \sum_i X_{ik} \geq - d_k \ \forall k \tag{22}$$

$$\mathrm{Y_i - (1/K) \sum_k (X_{ik}/d_k) \geq 0,} \ \forall i \tag{23}$$

$$X_{ik} \geq 0, \ \forall i,k \tag{24}$$

$$Y_i = (0,1), \ \forall i \tag{25}$$

Now the weak relaxation of SPLP is as follows:

Problem RWP: min $\sum_i \sum_k X_{ik}C_{ik} + \sum_i f_i Y_i$

s.t eq. (21) to (24) and $Y_i \geq 0, \ \forall i.$

We associate dual variables v_0 with equations (21), v_k with equations (22) and w_i with equations (23). The dual of problem RWP is as follows

Problem DRWP: max $v_0 - \sum_k v_k \, d_k$

$$\text{s.t. } v_0 - v_k - w_i/(Kd_k) \leq C_{ik} \tag{26}$$

$$w_i \leq f_i \quad \forall i, \ v_k \geq 0, \forall k, \ w_i \geq 0, \forall i, \ v_0 \ \textit{unrestricted in sign.}$$

It can be easily seen that now we have an identical result as theorem 1 of section three; and as a consequence there exists an optimal solution to problem DRSP with $w_i = f_i, \forall i.$ which then reduces to a problem similar to problem DRP1 and its optimal solution is

$$\sum_{k} d_k \min_i (C_{ik} + f_i/(Kd_k)) \qquad (27)$$

The procedure for calculating exact solution to WRS is different from that given by Effroymson and Ray [8].

CHAPTER 3

RESEARCH PROBLEM

Problem 1

Wanted to see, if significant difference existed between the relaxations WRS(Old) and SRS and then compare both with WRS(New) given by Sharma & Muralidhar.

In chapter 2 during literature review we have discussed that the Sharma and Muralidhar has given new relaxation for Uncapacitated Plant Location Problem and claimed that the bounds given by new relaxation are easily available (in $O(n^2)$ time) and can be better than the bounds given by the "Weak Relaxation" of SPLP, depending upon the problem instances, however the bounds given by this new relaxation are worse than the bounds given by "Strong Relaxation" of SPLP. We have tried here to examine their claim and to explore trend, if any.

Problem 2

To know the efficiency of Erlenkotter's Dual Ascent procedure to estimate the bounds given by SRS (of SPLP) for problems of various kinds.

As we have discussed in chapter 2, that Erlenkotter's Dual Ascent procedure provides better bound for SPLP in very few iterations in comparison to SRS. We have tried here to find efficiency of Dual Ascent to provide better bound for SPLP and tried to find the region in which this procedure gives inferior results.

Problem 3

It was found by Sharma and Berry [ ] that Big-M formulation gives better bounds for SSCWLP, when compared with constraints that link location variables (y_j's; 0-1) and real variables (x_{ij}'s); to distribution variables we tried to see whether a similar strategy works to SPLP also.

Problem 4

Develop a new formulation and relaxation, for Capacitated Plant Location Problem (CPLP). In this we have tried new constraints for Capacitated Plant Location Problem and found that it provides better bounds, as in the case of Sharma and Berry [17]

CHAPTER 4

EFFICIENCY OF DUAL ASCENT

4.1 Introduction

This chapter focuses upon the efficacy of Erlenkotter's Dual Ascent procedure to solve the Simple Plant Location Problem. Dual Ascent procedure is thoroughly explained in chapter 2.

4.2 SRS (Strong relaxation for SPLP)

The following formulation is for SRS and has already been discussed in chapter 2

SRS: $$\text{Min} \sum_i \sum_k X_{ik} C_{ik} + \sum_i f_i Y_i$$

Subject To: $$\sum_i \sum_k X_{ik} = 1 \tag{1}$$

$$-\sum_i X_{ik} \geq -d_k \ \forall k \tag{2}$$

$$d_k Y_i - X_{ik} \geq 0 \ \forall i,k \tag{3}$$

$$X_{ik} \geq 0, \quad \forall i,k \tag{4}$$

$$Y_i \geq 0, \quad \forall i \tag{5}$$

4.3 Details about Empirical Investigation

Here we have tried to find the efficiency of Dual Ascent Procedure. We have tried 3 problems each for problems set of 10x10, 20x20, 30x30, 40x40, 50x50, 60x60, 70x70. Here 10x10 means that problem is having 10 markets and 10 plants and tried each problem for $f(i)$ = 0, 1000, 2000, 3000, 5000, 10000, 20000, 50000, 100000, 200000, 500000, 1000000, 5000000, 8000000, 10000000 $\forall i$. here 'i' is for plant.

The values for c_{ik} and d_k were randomly generated through a program.

We have programmed the Dual Ascent procedure in 'C' language, which is given in Appendix (A) and the we solve SRS with Lingo Software. Results are shown in Appendix (B) and the Lingo Formulation for SRS is given in Appendix (C).

The tables and graphs generated are enclosed at the end of this chapter.

4.3 Results and Discussion

It can be seen in majority of problems that the difference between the bounds for LP given by Lingo formulation for SRS and those given by Erlenkotter's Dual Ascent, is a closed curve which starts and ends at zero and in-between, the difference varies and goes up to 9% (in some cases). This shows that Dual Ascent procedure due to Erlencotter [9] for SPLP is a good solution procedure to estimate the bounds given by SRS formulation.

Table 4.1 Showing the Max. Percentage Difference and Range for Gap in SRS & Dual Ascent for each Problem Size

Problem Size	Max Gap in SRS & Dual Ascent when Ratio of $\Sigma f(i)*y(i)$ to SRS Obj. Function Value(in %) is	Max. Percentage Difference
10x10	30%-50%	4.37%
20x20	30%-40%	7.35%
30x30	30%-60%	6.41%
40x40	30%-70%	5.81%
50x50	30%-100%	7.51%
60x60	30%-90%	6.64%
70x70	30%-70%	6.97%

Here f(i) is fixed cost for establishing facility at 'i'th location and y(i) is a variable showing whether 'i'th facility is selected or not.

Percentage difference *is the difference between Objective function value of SRS and of Dual Ascent procedure,* (in *%*)
i.e. 1-(*obj. func. value of Dual Ascent*) / *value of SRS* , in %

Ratio is $\Sigma f(i)*y(i)/$*Obj. func. value of SRS*

4.4 Inferences

It is thus seen that gap between SRS & Dual Ascent due to Erlencotter is max when ratio is higher than 30% and the range grows higher as problem size increases.

Fig. 4.1 Difference Between Dual Ascent & SRS (10x10)

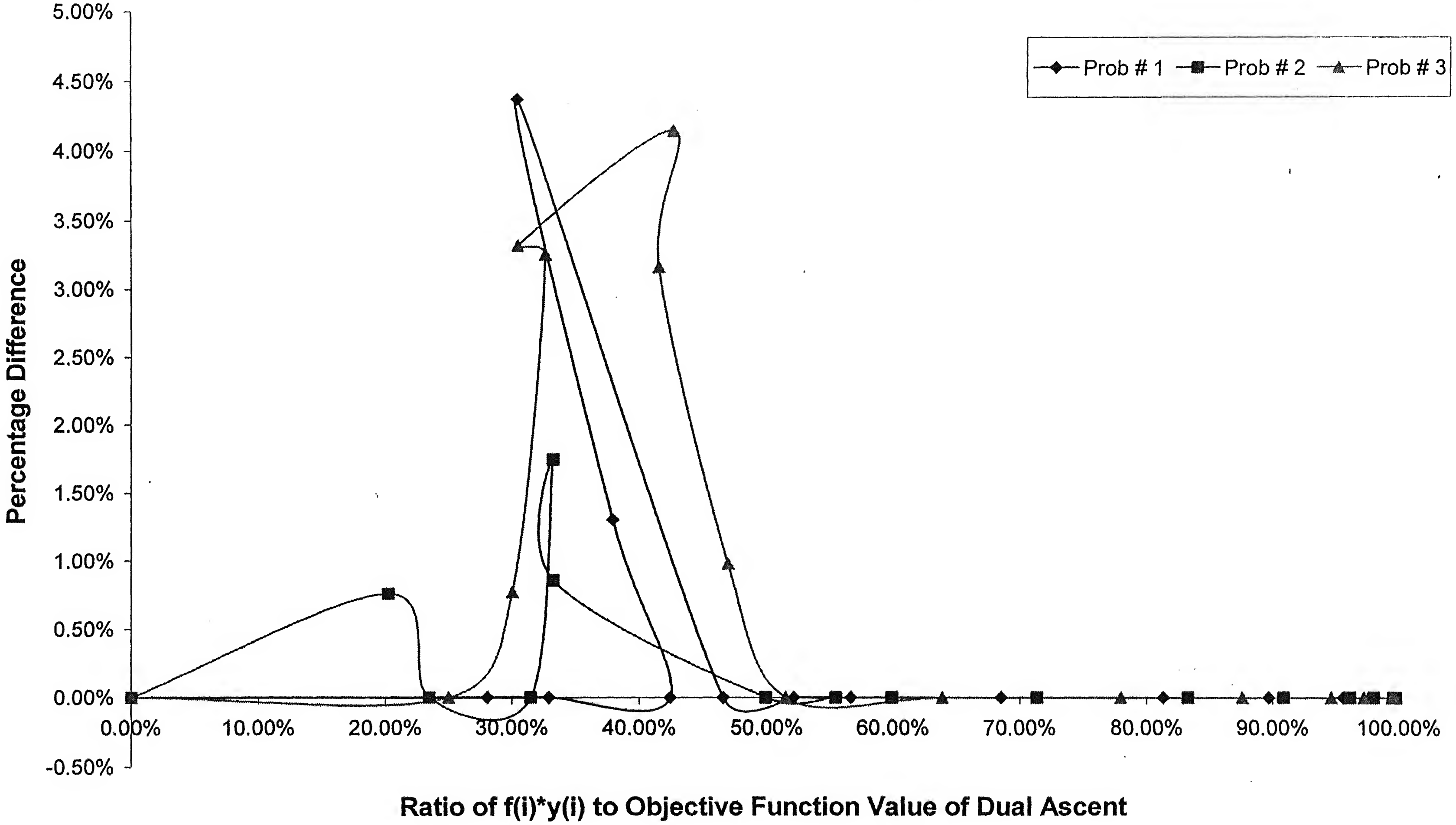


Fig. 4.2 Difference Between Dual Ascent & SRS (20x20)

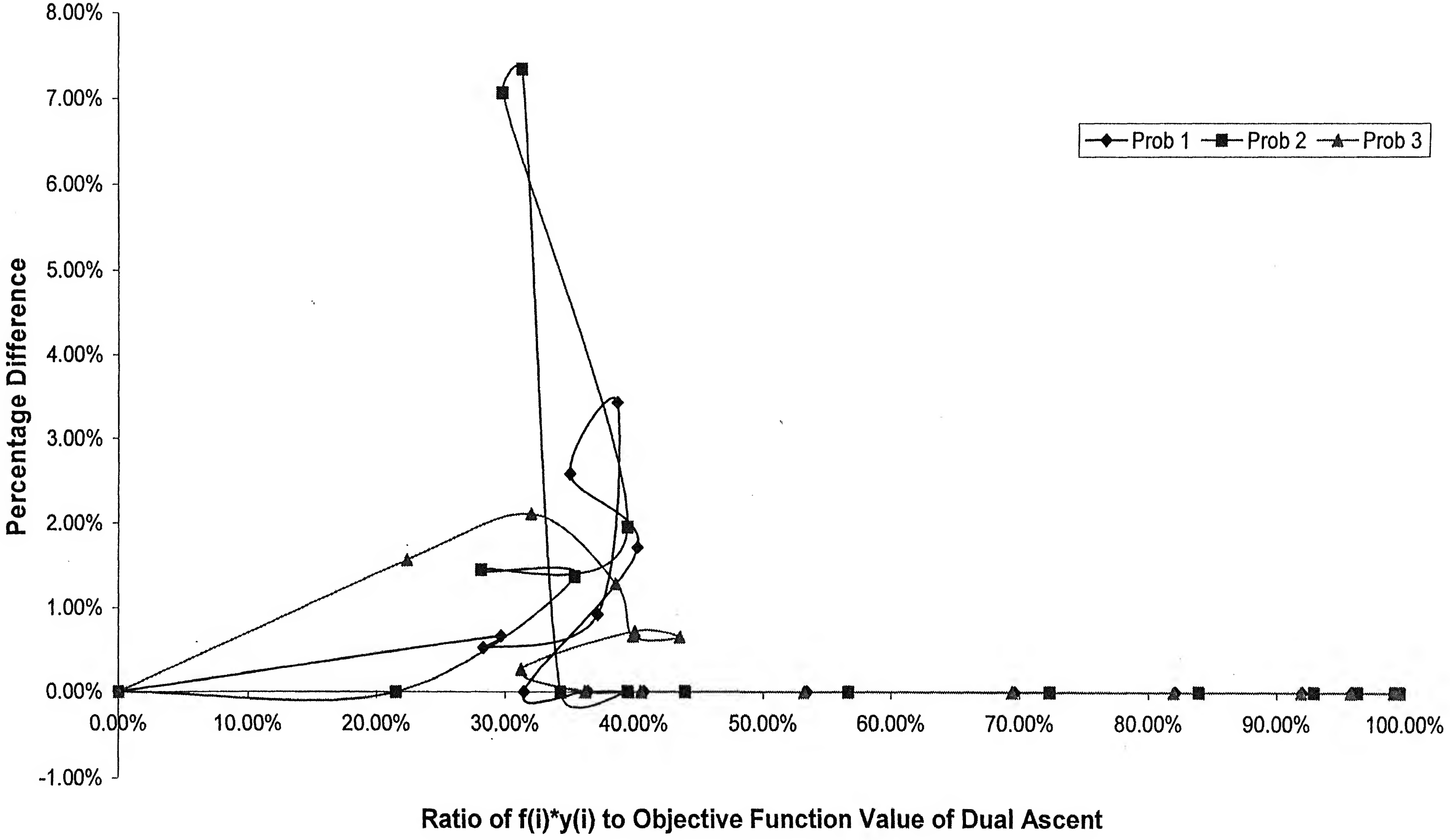


Fig. 4.3 Difference Between Dual Ascent & SRS (30x30)

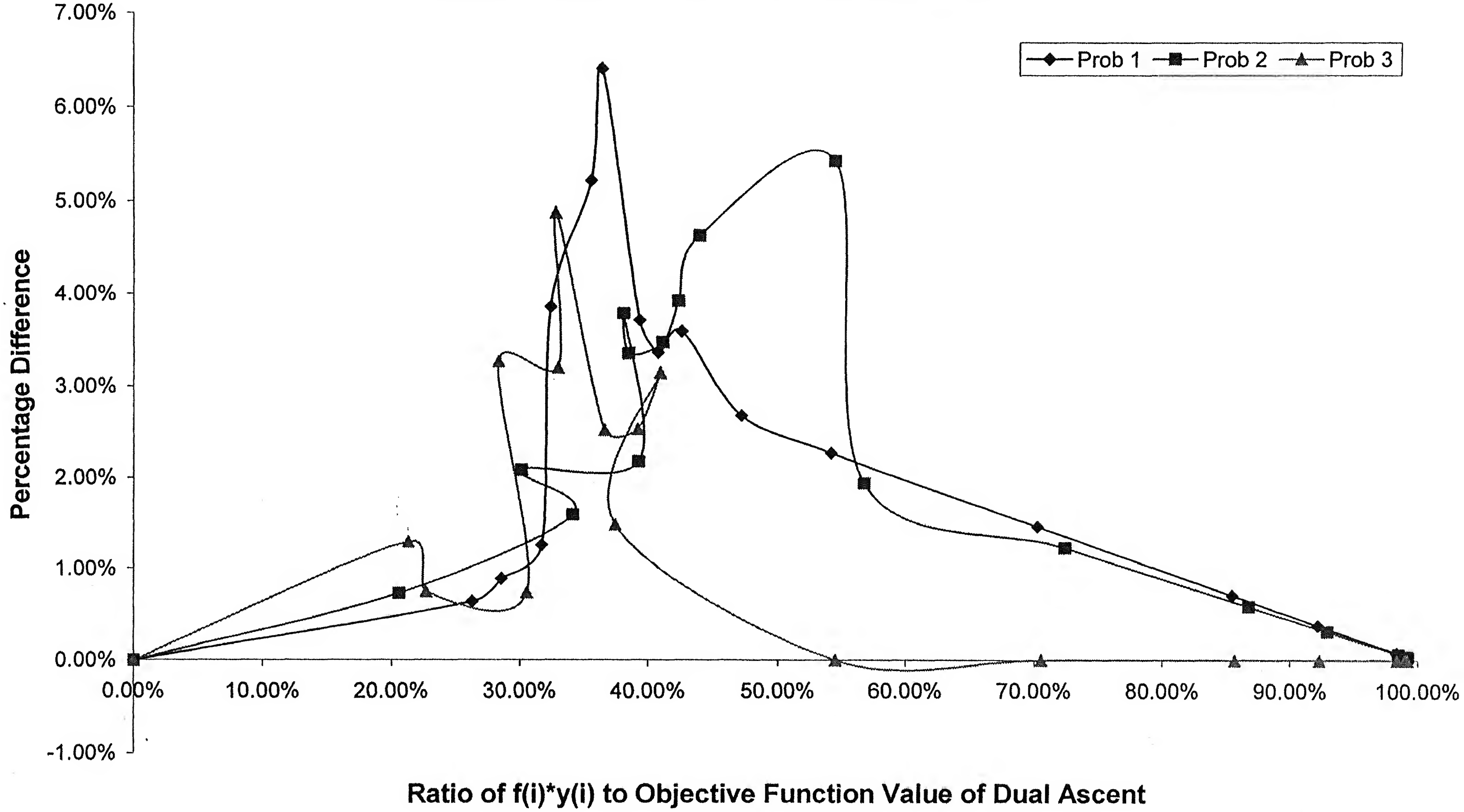


Fig 4.4 Difference Between Dual Ascent & SRS (40x40)

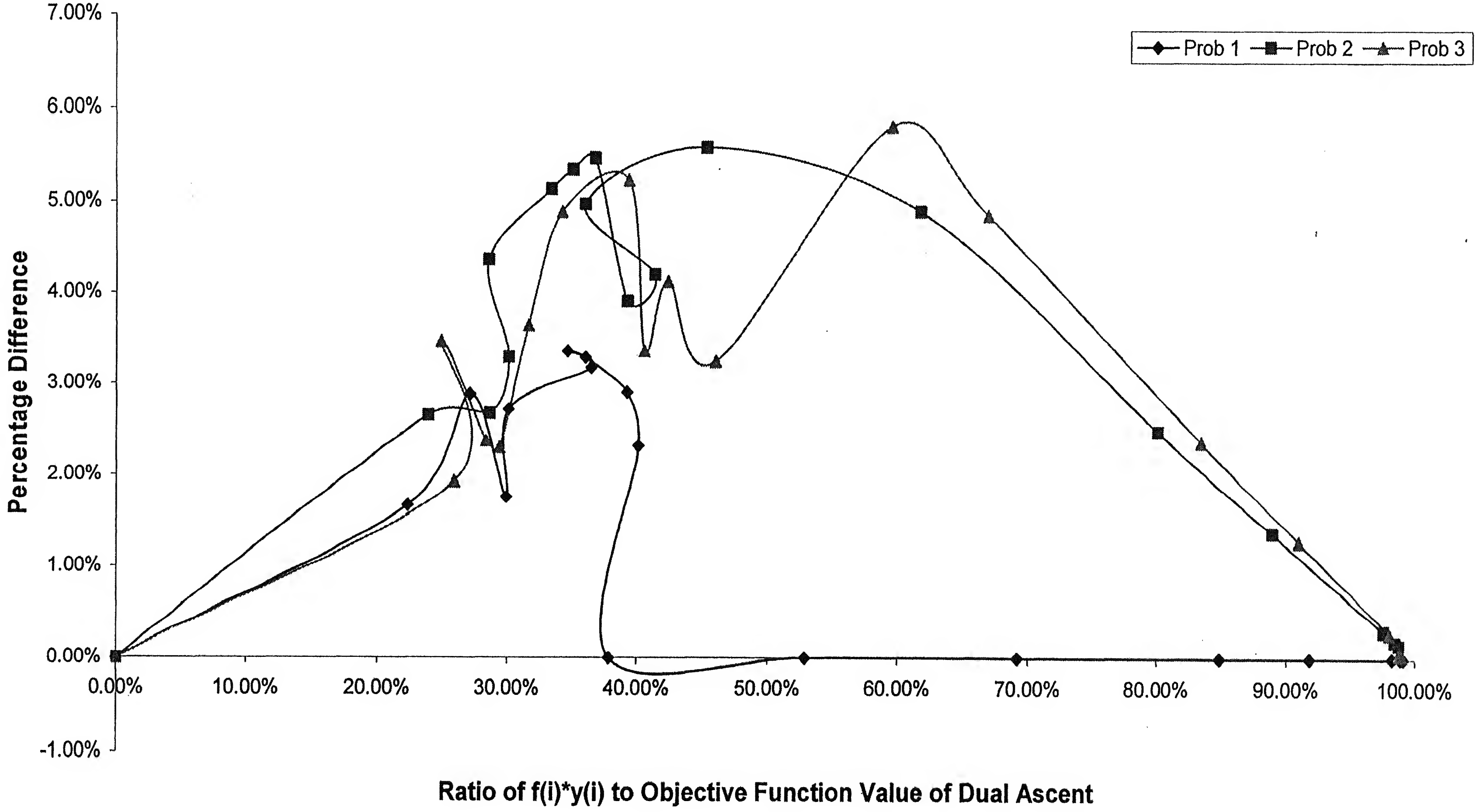


Fig. 4.5 Difference Between Dual Ascent & SRS (50x50)

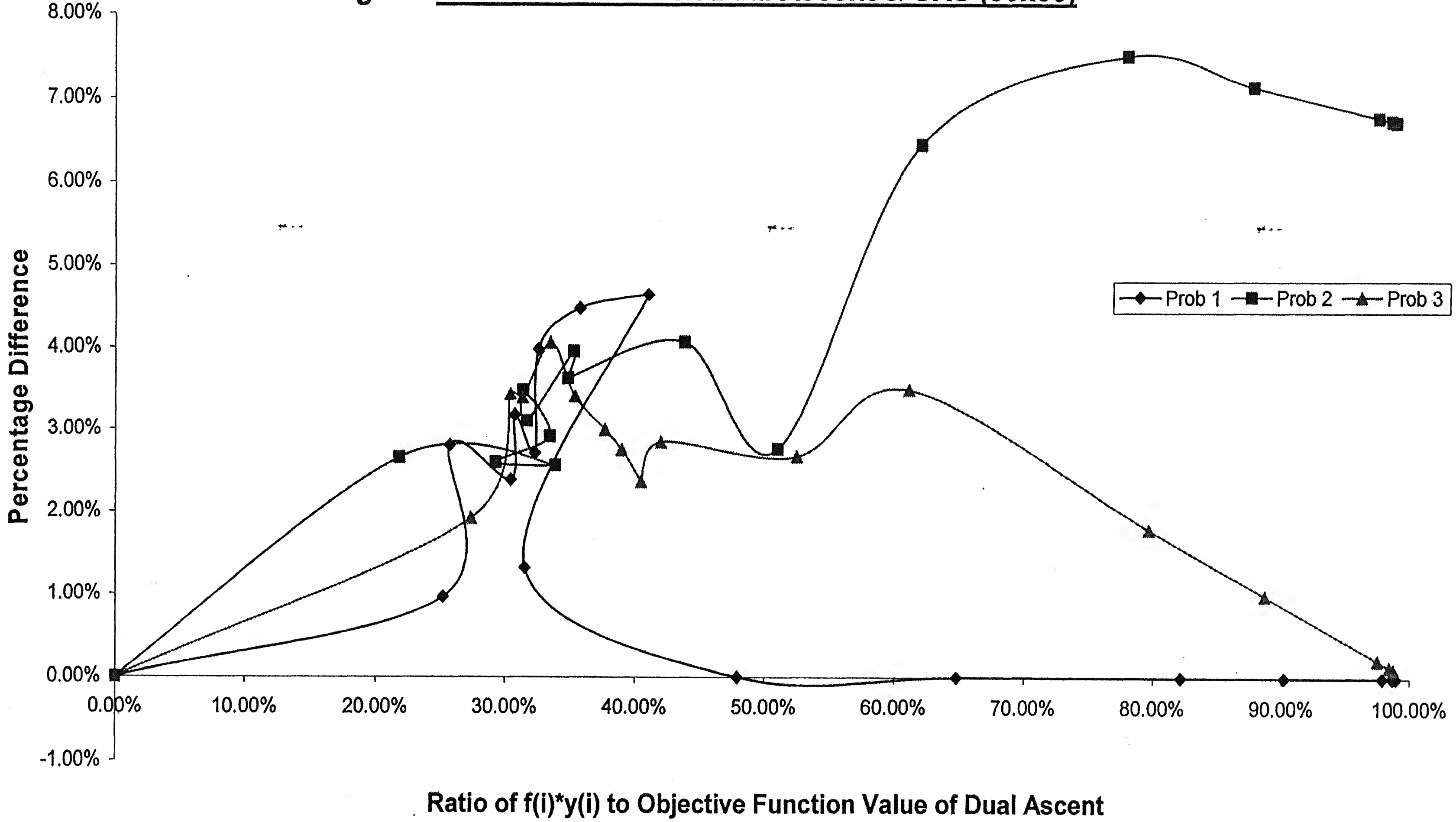


Fig. 4.6 Difference Between Dual Ascent & SRS (60x60)

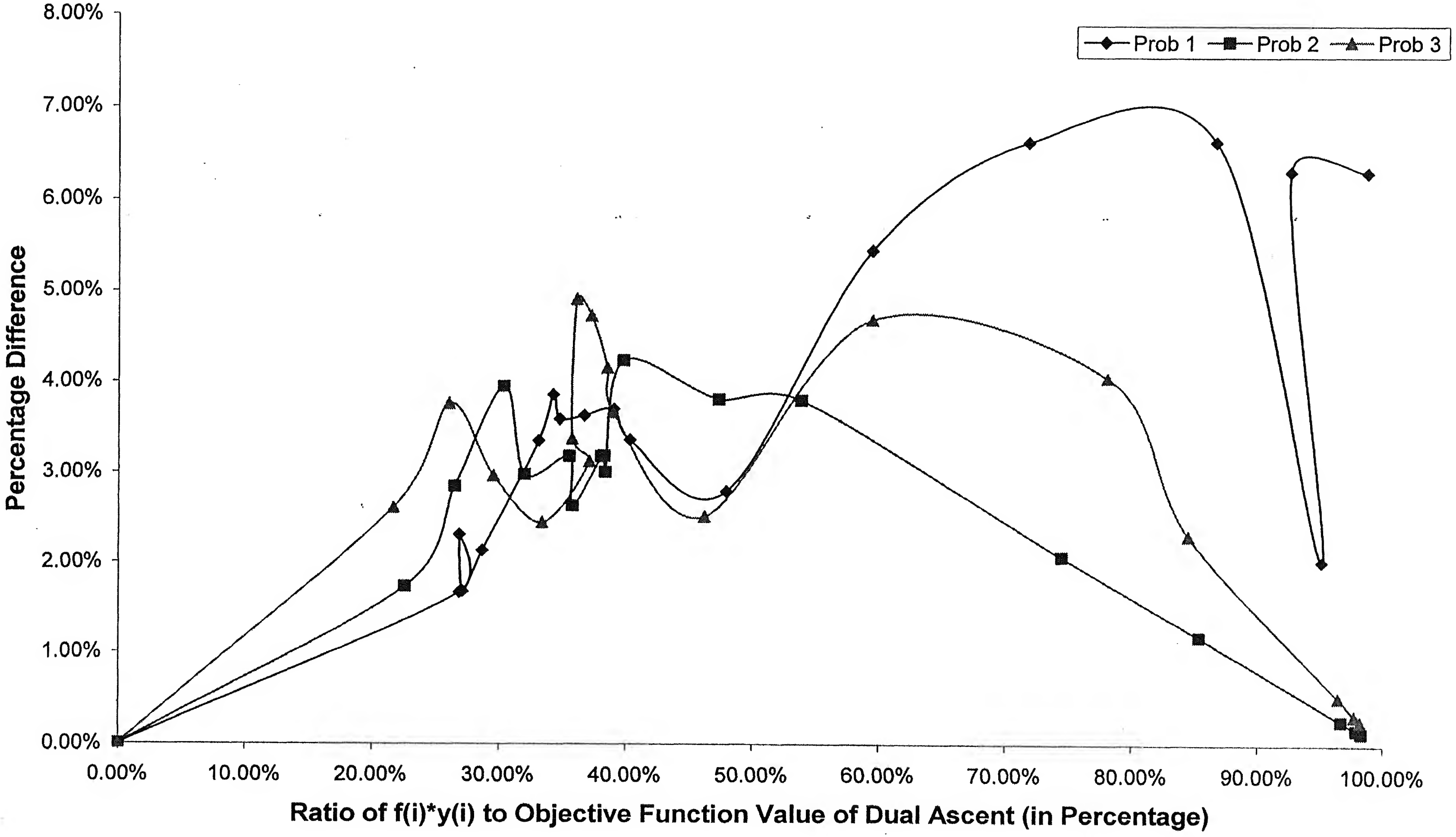
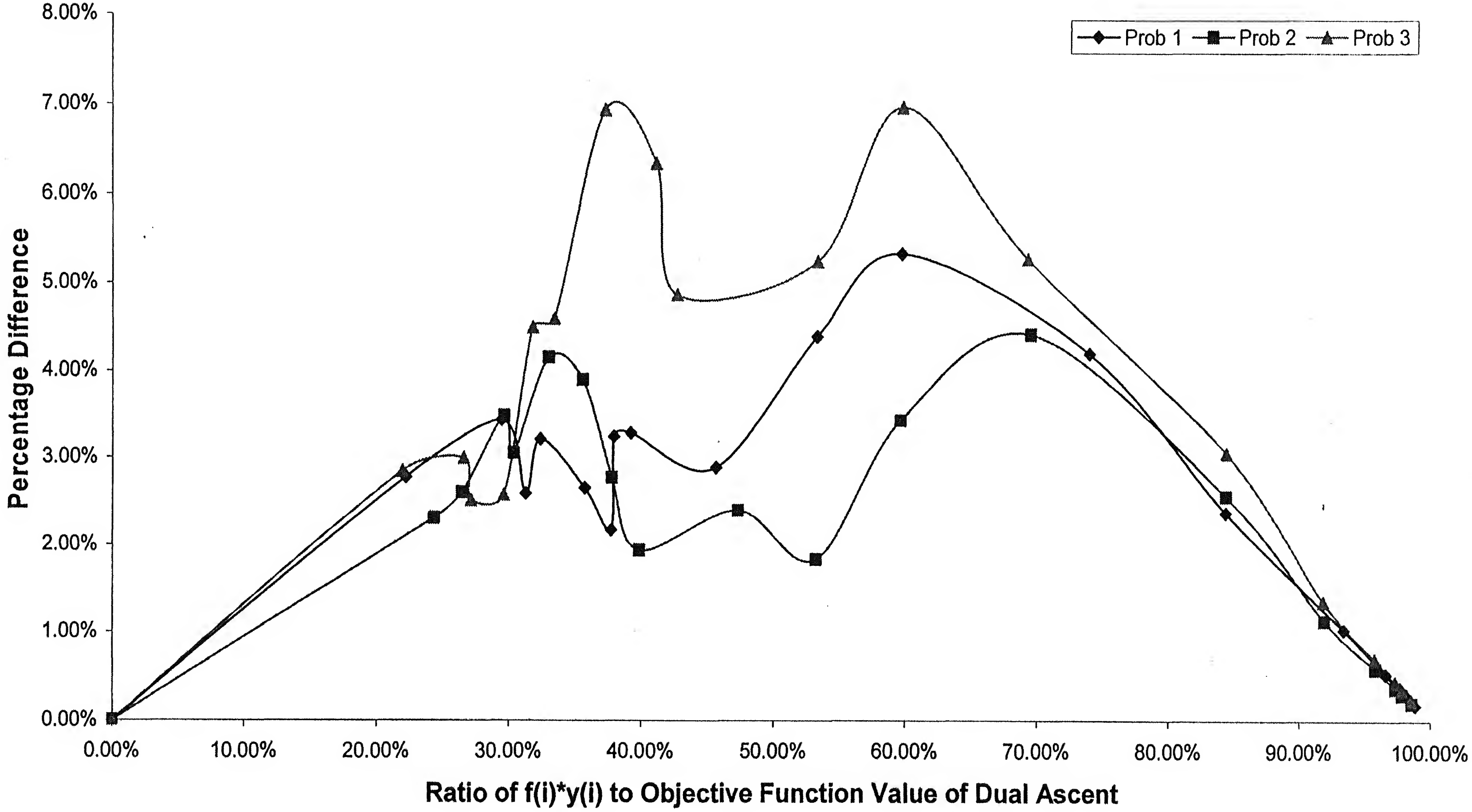


Fig. 4.7 Difference Between Dual Ascent & SRS (70x70)



Table#4.2. *Percentage Difference and Ratio of f(i)y(i) to Obj. Function value of SRS For Problem Set 10x10*

Value of f(i)	Problem # 1		Problem # 2		Problem # 3	
	Perc. Diff.	Ratio	Perc. Diff.	Ratio	Perc. Diff.	Ratio
0	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
1000	0.00%	28.06%	0.76%	20.23%	0.00%	25.00%
2000	0.00%	32.92%	0.00%	23.46%	0.77%	30.04%
3000	0.00%	42.40%	0.00%	31.49%	3.25%	32.56%
5000	1.30%	37.93%	1.75%	33.22%	3.32%	30.42%
10000	4.37%	30.35%	0.86%	33.26%	4.15%	42.54%
20000	0.00%	46.57%	0.00%	49.92%	3.16%	41.43%
25000	0.00%	52.14%	0.00%	55.48%	0.98%	46.93%
30000	0.00%	56.66%	0.00%	59.92%	0.00%	51.48%
50000	0.00%	68.54%	0.00%	71.36%	0.00%	63.88%
100000	0.00%	81.34%	0.00%	83.29%	0.00%	77.96%
200000	0.00%	89.71%	0.00%	90.88%	0.00%	87.61%
500000	0.00%	95.61%	0.00%	96.14%	0.00%	94.65%
1000000	0.00%	97.76%	0.00%	98.03%	0.00%	97.25%
5000000	0.00%	99.54%	0.00%	99.60%	0.00%	99.44%
8000000	0.00%	99.71%	0.00%	99.75%	0.00%	99.65%
10000000	0.00%	99.77%	0.00%	99.80%	0.00%	99.72%

Table#4.3 *Percentage Difference and Ratio of f(i)y(i) to Obj. Function value of SRS For Problem Set 20x20*

Value of f(i)	Problem # 1		Problem # 2		Problem # 3	
	Perc. Diff.	Ratio	Perc. Diff.	Ratio	Perc. Diff.	Ratio
0	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
1000	0.67%	29.69%	0.00%	21.52%	1.57%	22.36%
2000	0.53%	28.32%	1.37%	35.41%	2.12%	32.02%
3000	0.93%	37.21%	1.45%	28.13%	1.29%	38.59%
5000	3.44%	38.67%	1.96%	39.48%	0.67%	39.92%
8000	2.60%	35.04%	7.07%	29.68%	0.65%	43.56%
10000	1.72%	40.27%	7.35%	31.20%	0.73%	40.06%
20000	0.00%	31.47%	0.00%	34.34%	0.27%	31.24%
25000	0.00%	36.46%	0.00%	39.53%	0.00%	36.23%
30000	0.00%	40.78%	0.00%	43.96%	0.00%	40.53%
50000	0.00%	53.44%	0.00%	56.66%	0.00%	53.18%
100000	0.00%	69.66%	0.00%	72.33%	0.00%	69.44%
200000	0.00%	82.11%	0.00%	83.95%	0.00%	81.96%
500000	0.00%	91.99%	0.00%	92.89%	0.00%	91.91%
1000000	0.00%	95.83%	0.00%	96.32%	0.00%	95.78%

5000000	0.00%	99.14%	0.00%	99.24%	0.00%	99.13%
8000000	0.00%	99.46%	0.00%	99.52%	0.00%	99.45%
10000000	0.00%	99.57%	0.00%	99.62%	0.00%	99.56%

Table#4.4 Percentage Difference and Ratio of f(i)y(i) to Obj. Function value of SRS
For Problem Set 30x30

Value of *f(i)*	*Problem # 1*		*Problem # 2*		*Problem # 3*	
	Perc. Diff.	Ratio	Perc. Diff.	Ratio	Perc. Diff.	Ratio
0	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
1000	0.64%	26.27%	0.73%	20.59%	1.29%	21.29%
2000	0.89%	28.55%	1.59%	34.15%	0.74%	22.68%
3000	1.26%	31.71%	2.08%	30.12%	0.74%	30.55%
5000	3.87%	32.36%	2.18%	39.29%	3.27%	28.29%
8000	5.22%	35.50%	3.79%	38.06%	3.20%	32.95%
10000	6.41%	36.35%	3.36%	38.46%	4.88%	32.73%
20000	3.72%	39.31%	3.48%	41.08%	2.52%	36.61%
25000	3.37%	40.73%	3.93%	42.28%	2.53%	39.20%
30000	3.60%	42.56%	4.64%	43.87%	3.15%	40.91%
50000	2.68%	47.16%	5.43%	54.36%	1.48%	37.47%
100000	2.27%	54.14%	1.94%	56.75%	0.00%	54.51%
200000	1.46%	70.25%	1.23%	72.41%	0.00%	70.56%
500000	0.71%	85.51%	0.59%	86.77%	0.00%	85.70%
1000000	0.38%	92.19%	0.31%	92.92%	0.00%	92.30%
5000000	0.08%	98.33%	0.07%	98.50%	0.00%	98.36%
8000000	0.05%	98.95%	0.04%	99.06%	0.00%	98.97%
10000000	0.04%	99.16%	0.03%	99.24%	0.00%	99.17%

Table#4.5 Percentage Difference and Ratio of f(i)y(i) to Obj. Function value of SRS
For Problem Set 40x40

Value of *f(i)*	*Problem # 1*		*Problem # 2*		*Problem # 3*	
	Perc. Diff.	Ratio	Perc. Diff.	Ratio	Perc. Diff.	Ratio
0	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
1000	1.67%	22.30%	2.65%	23.86%	1.93%	25.88%
2000	2.88%	27.08%	2.67%	28.61%	3.47%	24.83%
3000	1.76%	29.91%	3.30%	30.08%	2.37%	28.34%
5000	2.72%	30.07%	4.36%	28.48%	2.31%	29.39%
8000	3.19%	36.47%	5.13%	33.31%	3.64%	31.60%
10000	3.36%	34.63%	5.35%	34.99%	4.89%	34.15%
20000	3.29%	36.03%	5.47%	36.71%	5.23%	39.26%
25000	2.91%	39.20%	3.91%	39.22%	3.37%	40.52%
30000	2.32%	40.11%	4.20%	41.28%	4.13%	42.27%

50000	0.00%	37.88%	4.98%	35.95%	3.25%	45.95%
100000	0.00%	52.89%	5.58%	45.17%	5.81%	59.35%
200000	0.00%	69.19%	4.89%	61.57%	4.85%	66.75%
500000	0.00%	84.88%	2.49%	80.02%	2.37%	83.38%
1000000	0.00%	91.82%	1.37%	88.90%	1.28%	90.94%
5000000	0.00%	98.25%	0.30%	97.56%	0.27%	98.05%
8000000	0.00%	98.90%	0.19%	98.46%	0.05%	98.77%
10000000	0.00%	99.12%	0.15%	98.77%	0.04%	99.01%

Table#4.6 Percentage Difference and Ratio of f(i)y(i) to Obj. Function value of SRS
For Problem Set 50x50

Value of *f(i)*	*Problem # 1*		*Problem # 2*		*Problem # 3*	
	Perc. Diff.	Ratio	Perc. Diff.	Ratio	Perc. Diff.	Ratio
0	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
1000	0.96%	25.22%	2.66%	21.80%	1.92%	27.36%
2000	2.81%	25.70%	2.57%	33.84%	3.44%	30.37%
3000	2.39%	30.42%	2.60%	29.26%	3.39%	31.31%
5000	3.18%	30.71%	2.92%	33.41%	4.06%	33.45%
8000	2.71%	32.28%	3.48%	31.34%	3.41%	35.34%
10000	3.98%	32.50%	3.11%	31.64%	3.00%	37.65%
20000	4.48%	35.67%	3.96%	35.20%	2.76%	38.93%
25000	4.65%	40.93%	3.63%	34.80%	2.37%	40.43%
30000	4.65%	40.93%	3.63%	34.80%	2.37%	40.43%
50000	1.32%	31.53%	4.07%	43.74%	2.85%	41.95%
100000	0.00%	47.95%	2.77%	51.01%	2.68%	52.47%
200000	0.00%	64.82%	6.45%	61.83%	3.49%	61.05%
500000	0.00%	82.16%	7.51%	77.69%	1.79%	79.67%
1000000	0.00%	90.21%	7.14%	87.44%	0.99%	88.68%
5000000	0.00%	97.88%	6.77%	97.20%	0.22%	97.51%
8000000	0.00%	98.66%	6.73%	98.24%	0.14%	98.43%
10000000	0.00%	98.93%	6.72%	98.58%	0.11%	98.74%

Table#4.7 Percentage Difference and Ratio of f(i)y(i) to Obj. Function value of SRS
For Problem Set 60x60

Value of *f(i)*	*Problem # 1*		*Problem # 2*		*Problem # 3*	
	Perc. Diff.	Ratio	Perc. Diff.	Ratio	Perc. Diff.	Ratio
0	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
1000	1.66%	26.91%	1.73%	22.60%	2.61%	21.68%
2000	2.30%	26.88%	2.85%	26.52%	3.76%	26.02%
3000	1.67%	27.15%	3.95%	30.35%	2.96%	29.54%
5000	2.13%	28.69%	2.98%	31.98%	2.44%	33.39%

8000	3.35%	33.09%	3.18%	35.50%	3.13%	37.08%
10000	3.85%	34.20%	2.63%	35.76%	3.37%	35.71%
20000	3.59%	34.74%	3.18%	38.04%	4.91%	36.02%
25000	3.63%	36.68%	3.19%	38.23%	4.73%	37.18%
30000	3.70%	38.99%	3.01%	38.34%	4.16%	38.46%
50000	3.36%	40.29%	4.24%	39.73%	3.66%	38.92%
100000	2.80%	47.93%	3.81%	47.29%	2.51%	46.21%
200000	5.45%	59.27%	3.80%	53.79%	4.69%	59.34%
500000	6.63%	71.53%	2.07%	74.43%	4.05%	77.93%
1000000	6.64%	86.33%	1.18%	85.34%	2.30%	84.43%
5000000	2.02%	95.06%	0.26%	96.68%	0.52%	96.44%
8000000	6.31%	92.27%	0.17%	97.90%	0.32%	97.75%
10000000	6.30%	98.42%	0.13%	98.31%	0.26%	98.19%

Table#4.8 Percentage Difference and Ratio of f(i)y(i) to Obj. Function value of SRS
For Problem Set 70x70

Value of *f(i)*	*Problem # 1*		*Problem # 2*		*Problem # 3*	
	Perc. Diff.	Ratio	Perc. Diff.	Ratio	Perc. Diff.	Ratio
0	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
1000	2.77%	22.20%	2.31%	24.32%	2.85%	21.92%
2000	3.43%	29.42%	2.60%	26.45%	2.99%	26.56%
3000	2.59%	31.25%	3.48%	29.61%	2.51%	27.12%
5000	3.21%	32.34%	3.05%	30.33%	2.57%	29.59%
8000	2.65%	35.71%	4.15%	32.94%	4.49%	31.72%
10000	2.17%	37.74%	3.90%	35.54%	4.59%	33.35%
25000	3.24%	37.91%	2.77%	37.76%	6.94%	37.13%
50000	3.28%	39.19%	1.94%	39.85%	6.34%	41.00%
100000	2.89%	45.60%	2.40%	47.29%	4.86%	42.62%
150000	4.39%	53.20%	1.83%	53.20%	5.24%	53.20%
200000	5.33%	59.57%	3.43%	59.57%	6.97%	59.57%
500000	4.19%	73.82%	4.41%	69.36%	5.27%	69.09%
1000000	2.37%	84.30%	2.56%	84.30%	3.05%	84.30%
5000000	0.53%	96.57%	0.59%	95.77%	0.70%	95.72%
8000000	0.33%	97.83%	0.37%	97.31%	0.44%	97.28%
10000000	0.27%	98.26%	0.30%	97.84%	0.36%	97.81%

CHAPTER 5

COMPARISON OF DIFFERENT FORMULATIONS

5.1 Introduction

In this chapter we have compared the bounds given by the old weak formulation of SPLP (WRS Old) and the new weak formulation of SPLP (WRS New). Bounds from both the formulations then compared with the bounds from strong formulation of SPLP (SRS). All the SRS, WRS(Old/New) are already discussed in chapter 2.

5.2 WRS(New): First Formulation

as discussed in chapter 2

WRS: $$\min \sum_i \sum_k X_{ik}C_{ik} + \sum_i f_i Y_i$$

$$\text{s.t.} \quad \sum_i \sum_k X_{ik} = 1 \tag{1}$$

$$-\sum_i X_{ik} \geq -d_k \ \forall k \tag{2}$$

$$\mathrm{Y_i} \geq \sum_k \mathrm{X_{ik}}, \ \forall\, i=1..I \tag{3}$$

$$X_{ik} \ \geq 0, \forall i,k \tag{4}$$

$$Y_i \ \geq 0, \forall i \tag{5}$$

5.3 WRS(Old): Second Formulation

In lieu of constraint 3 when we take constraint 6 then it becomes WRS(Old)

$$Y_i - (1/K) \sum_k \ (X_{ik}/d_k) \geq 0, \ \forall \mathrm{i} \tag{6}$$

Then the problem becomes,

$$\text{Min: } \sum_i \sum_k X_{ik}C_{ik} + \sum_i f_i Y_i$$

Subject to: (1)-(2), (3)-(5) and (6)

5.4 SRS:

In lieu of constraint 3 when we take constraint 7 then it becomes SRS

$$d_kY_i - X_{ik} \geq 0 \,\forall i,k \tag{7}$$

Then the problem becomes,

$$\text{Min: } \sum_i \sum_k X_{ik}C_{ik} + \sum_i f_i Y_i$$

Subject to: (1)-(2), (3)-(5) and (7)

5.5 Details About The Empirical Investigation

In total we have solved 30 problems for 10x10 and 20x20 and for 30x30, 40x40, 50x50, 60x60, 70x70 we solved 15 problems for each category. Here 10x10 means 10 plants and 10 markets. So in total we have solved around 135 problems. We also have investigated the difference between both the relaxations with the SRS for the same problem. We performed T-test over the bounds given by each formulation and checked it for **mean of difference = 0**.

A code in 'C' language was written both for WRS Old and New to solve the problems, and SRS problems were solved through Lingo software. The Lingo formulation and 'C' code is given in Appendix (C).

As we have discussed in chapter 2, the values for c_{ik}, d_k, and f_i are randomly generated through a computer program encoded in JAVA language. Details about this program are given in Appendix.

The tables containing the bounds for each category is shown in Appendix. Here the table containing the results of t-test for each category is shown:

Table No.3 and 4 shows the value of t-critical for different DOF and for different values of α. For 10x10 and 20x20 the DOF is 29 and for rest, DOF is 14.

TABLE-5.1 T-Values For Different Formulations:

Hypothesis: Mean of Difference = 0

Formulations	*T-Value*	*Status of Hypo.*
10x10 First Formulation V/s Second formulation	**1.782224**	**Accepted(α=0.025)**
10x10 First Formulation V/s SRS	-20.5511	(Rejected)
10x10 Second Formulation V/s SRS	-20.1	(Rejected)
20x20 First Formulation V/s Second formulation	**1.790678**	**Accepted(α=0.025)**
20x20 First Formulation V/s SRS	-28.2084	(Rejected)
20x20 Second Formulation V/s SRS	-28.2188	(Rejected)
30x30 First Formulation V/s Second formulation	**2.352233**	**Accepted(α=0.01)**
30x30 First Formulation V/s SRS	-21.5365	(Rejected)
30x30 Second Formulation V/s SRS	-21.8436	(Rejected)
40x40 First Formulation V/s Second formulation	**1.04815**	**Accepted(α=0.10)**
40x40 First Formulation V/s SRS	-30.5454	(Rejected)
40x40 Second Formulation V/s SRS	-29.8527	(Rejected)
50x50 First Formulation V/s Second formulation	**1.184483**	**Accepted(α=0.10)**
50x50 First Formulation V/s SRS	-45.7508	(Rejected)
50x50 Second Formulation V/s SRS	-32.1936	(Rejected)

60x60 First Formulation V/s Second formulation	**2.723534**	**Accepted(α=0.005)**
60x60 First Formulation V/s SRS	-42.6262	(Rejected)
60x60 Second Formulation V/s SRS	-42.1344	(Rejected)
70x70 First Formulation V/s Second formulation	**2.70134**	**Accepted(α=0.005)**
70x70 First Formulation V/s SRS	-29.3789	(Rejected)
70x70 Second Formulation V/s SRS	-29.44	(Rejected)

TABLE-5.2 Overall t-values

First Formulation V/s Second formulation	**2.068454**	**Accepted(α=0.005)**
First Formulation V/s SRS	-23.1186	(Rejected)
Second Formulation V/s SRS	-23.0847	(Rejected)

TABLE-5.3 ***T-critical For One Tail T-Test***

Significance level	**α = 0.10**	**α = 0.05**	**α = 0.025**	**α = 0.01**	**α = 0.005**
T-Critical (DOF-14)	1.345	1.761	2.145	2.624	2.977
T-Critical (DOF-29)	1.311	1.699	2.045	2.462	2.756
T-Critical (DOF-134)	1.311	1.699	2.045	2.462	2.756

TABLE-5.4 ***T-critical For Two Tails T-Test***

Significance Level	**α = 0.20**	**α = 0.10**	**α = 0.05**	**α = 0.02**	**α = 0.01**
T-Critical (DOF-14)	1.345	1.761	2.145	2.624	2.977
T-Critical (DOF-29)	1.311	1.699	2.045	2.462	2.756
T-Critical (DOF-134)	1.311	1.699	2.045	2.462	2.756

5.3 Results and Discussion

The tables containing t-values for each category is shown above and corresponding bounds are shown in Appendix (D). As from the details given in tables and the from the results of T-test, this can easily be seen that the difference between the WRS(Old) and WRS(New) is not significant with no clear trend, but the difference between SRS and WRS(Old/New) is significant and having positive difference. This shows that the bounds given by the WRS(Old) and WRS(New) are not having significant difference but the bounds given by SRS are better as having positive difference from both WRS(Old) and WRS(New).

CHAPTER 6

FORMULATION OF SPLP WITH BIG 'M'

6.1 Introduction:

In this chapter, we have relaxed few variations of model constraints and included some other linking constraints. The performance and behavior of the formulation is discussed here.

6.2 Mathematical Formulation:

We are including this new constraint to the formulation of SRS, WRS Old and New (Discussed in previous chapters).

$$\sum_k x_{ik} - M(1-y_i) \leq 0 \;\; \forall\, i = 1 \,..\, \mathrm{I}$$

$$\sum_k x_{ik} - M(y_i) \leq 0 \;\; \forall\, i = 1 \,..\, \mathrm{I}$$

$$\sum_k x_{ik} + M(y_i) \geq 0 \;\; \forall\, i = 1 \,..\, \mathrm{I}$$

This equation is another form of the linking constraint and ensures that the plant is located only when there is quantity shipped through the plant is greater than zero. We call this constraint the Big M constraint, where M is a constant having very large value.

We have tried this Big 'M' formulation in WRS (Old/New) and SRS and solved the randomly generated problems for 60x60 and 70x70, 15 problems for each, with value of M = 1000000. The Lingo formulation is shown in Appendix (C).

6.3 Results and Discussions

On the analysis of the results, we see that although there is no improvement in the bounds given by the new formulation over the bounds given by WRS (Old/New) and SRS. Also the number of iterations needed to solve the problems, are not having any trend, neither positive nor negative. Results are given in Appendix (G).

6.4 Inference

Thus we have established that there is no significant difference in the bounds given by the formulations without Big M and the formulations with Big M. This is because, when we include the constraints with Big M, it doesn't makes any difference to the bounds since the linking constraints are also doing the same thing over there, so in Simple Plant Location Problems this Big 'M' constraint is failed to make significant difference in the performance of formulations.

CHAPTER 7

EMPIRICAL INVESTIGATION OF CAPACITATED PLANT LOCATION PROBLEM, ITS RELAXATION AND ITS PERFORMANCE WITH BIG 'M'

7.1 Introduction:

In this chapter, we are discussing few formulations of Strong Relaxation for CPLP (SRS) and then we will compare these with some new formulations and its variations. In previous chapters we have investigated SRS of SPLP with Big 'M' and have seen that the bounds are not significantly different when Big 'M' is included in that. Here we are going to try the same with the plants having fixed capacity.

7.2 Constants Definition:

D_k Demand for the commodity at market '*k*'

d_k $D_k/\Sigma D_k$ demand at market '*k*' as a fraction of total market demand.

S_i Supply available at plant '*i*' of the commodity

s_i $S_i/\Sigma D_k$ supply available at plant '*i*' as a fraction of the total market demand

f_i Fixed cost of locating a plant at '*i*'.

C_{ijk} Cost of transporting $\sum_{k=1}^{K} D_k$ units of goods from '*i*' market '*k*' i.e. cost of transporting total market demand (of all markets) from plant *i* to market *k*.

7.3 Variable Definition:

X_{ik} quantity of commodity transported from plant '*i*' to market '*k*'

x_{ik} $X_{ik}/\sum D_k$ quantity transported as a fraction of total market demand

y_i 1 if plant is located at '*i*', 0 otherwise

7.4 Mathematical Formulation:

Now cost minimization problem for the CPLP can be written as the mixed integer programming problem as:

CPLP: $$\text{Min} \sum_{i,k} x_{ik} c_{ik} + \sum_{i} f_i y_i$$

s.t

$$\sum_{i,k} x_{ik} = 1 \tag{1}$$

$$-\sum_{i} x_{ik} \geq -d_k \tag{2}$$

$$\sum_{k} x_{ik} \leq s_i \tag{3}$$

$$x_{ik} \leq y_i d_k \tag{4}$$

$$x_{ik} \geq 0 \tag{5}$$

$$y_i = (0,1) \forall i \tag{6}$$

So the optimal integer solution to the problem is obtained by solving the above mixed integer formulation.

The first Part of Objective Function denotes the cost of transportation of the commodity from the plant to the market; the second part is fixed cost of locating the warehouse.

7.4.1 Strong Relaxation of CPLP (SRS or Relaxation P1):

Minimize (0) subject to (1) to (5) and

$$y_i \geq 0 \forall i$$

This can be referred as the strong relaxation as termed so in the literature.

7.4.2 Relaxation P2:

When we include Big 'M' constraint (7) with the previous one the problem becomes

Minimize (0) subject to (1)-(2),(4)-(5), $y_i \geq 0 \forall i$, and

$$\left.\begin{array}{l} \sum_k x_{ik} - M(1-y_i) \leq s_i \\ \sum_k xik + My_i \geq 0 \\ \sum_k x_{ik} - My_i \leq 0 \end{array}\right\} \tag{7}$$

This equation is another form of linking constraint and ensures that the plant is located only when there is quantity shipped through the plant is greater than zero. We call this constraint the Big 'M' constraint, where M is a constant having a very large value.

7.4.3 Strong Relaxation for CPLP with cap$_i$ (Relaxation P3):

Minimize (0) subject to (1)-(2),(4)-(5), $y_i \geq 0 \forall i$, and

$$\sum_k x_{ik} \leq s_i y_i \tag{8}$$

Equation is another form of linking constraints and also ensures that the plant is located only if quantity shipped through the warehouse is greater than zero. They relate the quantity shipped from a plant '*i*' to the market '*k*'. This is another form of a strong relaxation.

7.4.4 Strong Relaxation for CPLP with cap$_i$ and Big 'M' (Relaxation P4):

Minimize (0) subject to (1)-(2),(4)-(5), $y_i \geq 0 \forall i$, (7) and (8)

7.5 Structure of Empirical Investigation Carried

We have investigated over here that whether the Big 'M' is making any significant difference in the bounds. For this, firstly we have compared the first two relaxed formulations discussed above and then the last two.

We have solved 60 problems of 30x30 type with $cap_i(\Sigma s_i) = 2.0$. These problems were randomly generated and solved on optimization package Lingo5.0. The lingo formulation and the results in form of tables are shown in Appendix (E). We then performed T-test on the sample and the results are shown in tables below.

Table-7.1 T-values for Different Pairs

Hypothesis: Mean of Difference = 0

Formulations	*T-Value*	*DOF*	*Status Of Hypo.*
Relaxation P1 V/s **Relaxation P2**	16.16219315	62	Rejected
Relaxation P3 V/s **Relaxation P4**	-13.59	60	Rejected
Relaxation P1 V/s **Relaxation P3**	-17.799	60	Rejected

Table-7.2 T-critical for one tail T-Test

Significance Level	**α = 0.05**
T-Critical (DOF=60)	1.670648544
T-Critical (DOF=62)	1.669804988

Table-7.3 T-critical for two tails T-Test

Significance Level	α=0.05
T-Critical (DOF=60)	2.000297172
T-Critical (DOF=62)	1.99896931

7.6 Results and Discussions:

The results shown in the tables above clearly indicate that in case of both the pairs the bounds given by the two formulations are significantly different. As the t-valve is positive for the first pair, this means that the first one is giving higher bounds to the other and as our problem is minimization problem and we are using these bounds in Branch and Bound procedure to get the optimum integer value, so for that purpose the bounds given by the Big 'M' are inferior and not of our interest.

The t-value for the second formulation is negative and this means that Big 'M' constraint is going to boost the bounds as these are going to use further to get the optimal integer solution to the problem.

In the last formulation, as the value is negative and thus means that the Relaxation P3 is giving better bounds to the bounds given by SRS (Relaxation P1), this is the **key finding** of this chapter.

7.7 Implications:

Thus in this chapter, we give two relaxation of CPLP that are giving stronger bounds than the bounds given by well known strong relaxation of CPLP (SRS).

CHAPTER 8

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

Summery of Findings:

(1) We showed in this thesis that new relaxation of SPLP, as given by Sharma and Muralidhar, is in fact another (new) weak Relaxation of SPLP; and the demonstrated by empirical investigation.

(2) We found that dual ascent procedure for computing bounds as given by SRS for SPLP due to Erlenkotter [9], is within 8 % of the optimal value. This was also demonstrated empirically.

(3) Sharma and Berry [17] has demonstrated that "Big 'M' Constraints" and constraints of type $x_{ijk} <= s_i y_j$ (also called "$x_{ij} <= \mathrm{cap}_j y_j$" constraint) when added to SSCWLP (Single Stage Capacitated Warehouse Location Problem) improves bounds. We took a lot from this and tried applying these constraints to problem SPLP. However it was found that these constraints failed to boost bounds for SPLP.

(4) However when constraints tried by Sharma and Berry [17], were added to problem CPLP, they yield bounds that were significantly greater than SRS of CPLP. This is a useful contribution of this thesis.

(5) Earlier Cornuejols et al [7] have given 12 relaxations of CPLP. We should now initiate work to see where the new relaxations P2 and P3 are placed as far as the strengths of various relaxations of CPLP are concerned.

APPENDIX (A)

A program in 'C' language, written for Erlenkotter's Dual Ascent Procedure to solve various problems of different sizes that are randomly generated.

```
#include <stdio.h>


        int i , k, c_i_k_for_e[100][100], w_i_k_for_e[100][100],
min_of_c_i_k_for_e[100][100];
        int remaining_resources_for_e[100], f_i_for_e[100], v_k_for_e[100];
        int increase_possible, counter;
        int v_k_can_go_up[100];
        int increase_possible_by_c_i_k_route, dual_ascent_can_go_on;

void sorting()
{
        int i1, k1, j, m, n, sorted_array[100][100];
        int found;
        for (k1 =1; k1 <=k; k1++)
        {
                sorted_array[1][k1] =c_i_k_for_e[1][k1];
                min_of_c_i_k_for_e[1][k1] =1;
                for (i1 =2; i1 <=i; i1++)
                {
                        sorted_array[i1][k1] =c_i_k_for_e[i1][k1];
                        min_of_c_i_k_for_e[i1][k1] =i1;
                        if(sorted_array[i1][k1] <sorted_array[i1-1][k1])
                        {
                                j =1;
                                found =1;
                                while(j <=(i1-1) && found==1)
                                {
                                        if (sorted_array[j][k1] >c_i_k_for_e[i1][k1])
                                                found =0;
                                        else
                                                j =j+1;
                                }
                        for(m=(i1-1); m>=j; m-- )
                        {
                                sorted_array[m+1][k1] =sorted_array[m][k1];
                                min_of_c_i_k_for_e[m+1][k1] =
min_of_c_i_k_for_e[m][k1];
                        }
                        sorted_array[j][k1] =c_i_k_for_e[i1][k1];
                        min_of_c_i_k_for_e[j][k1] =i1;
                        }
                }
        }

}

void find_increase_possible(int k1)
{
        int min_position, c_i_k_of_min_position, next_min_position;
        int c_i_k_of_next_min_position, increase_possible_from_c_i_k_route, i_value;
        int touching_index =0, i1 =0;

        increase_possible=0;
        increase_possible_by_c_i_k_route =1;
        touching_index =min_of_c_i_k_for_e[0][k1];
        min_position =min_of_c_i_k_for_e[touching_index][k1];
        c_i_k_of_min_position =c_i_k_for_e[min_position][k1];
        next_min_position =min_of_c_i_k_for_e[touching_index+1][k1];
        c_i_k_of_next_min_position =c_i_k_for_e[next_min_position][k1];
        if(c_i_k_of_min_position==c_i_k_of_next_min_position)
        {
                min_of_c_i_k_for_e[0][k1]++;
                touching_index++;
        }
        if(touching_index>=i)
        {
                return;
        }
        if(touching_index <1)
        {
                next_min_position =min_of_c_i_k_for_e[touching_index+1][k1];
                c_i_k_of_next_min_position =c_i_k_for_e[next_min_position][k1];
                increase_possible_from_c_i_k_route =c_i_k_of_next_min_position -
c_i_k_of_min_position;
                increase_possible_by_c_i_k_route =0;
                increase_possible =increase_possible_from_c_i_k_route;

                for(i1 =1; i1 <=touching_index; i1++)
                {
                        i_value =min_of_c_i_k_for_e[i1][k1];
                        if(remaining_resources_for_e[i_value] <increase_possible)
                        {
                                increase_possible_by_c_i_k_route =1;
                                increase_possible =remaining_resources_for_e[i_value];
                        }
                }
                for (i1 =1; i1 <=touching_index; i1 ++)
                {
                        i_value =min_of_c_i_k_for_e[i1][k1];
                        remaining_resources_for_e[i_value] =
remaining_resources_for_e[i_value]- increase_possible;
                }
        }
}

main()
{
                int i1, k1, objective_function_value_for_e, min_position, i_value;
                int touching_index =0;

                printf("array size please: ");
                scanf("%d%d", &i, &k);

                printf("enter array elements for f_i_for_e: ");
                for (i1 =1; i1 <=i; i1 ++)
                        scanf("%d", &f_i_for_e[i1]);
                        printf("\n");

                printf("\n enter array elements for c_i_k_for_e: ");
                for(i1 =1; i1 <=i; i1 ++)
                {
                                for(k1 =1; k1 <=k; k1 ++)
                                scanf("%d", &c_i_k_for_e[i1][k1]);
                        printf("\n");
                }

                for (i1 =1; i1 <=i; i1 ++)
                        for (k1 =1; k1 <=k; k1 ++)
                                w_i_k_for_e[i1][k1] =0;

                for (k1 =1; k1 <=k; k1 ++)
                        v_k_for_e[k1] =0;
                for (k1 =1; k1 <=k; k1 ++)
                        min_of_c_i_k_for_e[0][k1] =0;


                objective_function_value_for_e =0;
                sorting();
                for (k1 =1; k1 <=k; k1 ++)
                {
                        min_position =min_of_c_i_k_for_e[1][k1];
                        min_of_c_i_k_for_e[0][k1] =1;
                        v_k_for_e[k1] =c_i_k_for_e[min_position][k1];
                        objective_function_value_for_e =objective_function_value_for_e
+v_k_for_e[k1];
                }

                dual_ascent_can_go_on =0;
                for (k1 =1; k1 <=k; k1 ++)
                        v_k_can_go_up[k1] =0;
                for (i1 =1; i1 <=i; i1 ++)
                        remaining_resources_for_e[i1] =f_i_for_e[i1];

                while(dual_ascent_can_go_on ==0)
                {
                        for (k1 =1; k1 <=k; k1 ++)
                        {
                                if (v_k_can_go_up[k1] ==0)
                                        find_increase_possible(k1);
                                if (increase_possible_by_c_i_k_route ==0)
                                {
                                        min_of_c_i_k_for_e[0][k1] =
min_of_c_i_k_for_e[0][k1] +1;
                                        if(min_of_c_i_k_for_e[0][k1] ==i)
                                                v_k_can_go_up[k1] =1;
                                }
                                else
                                {
                                        v_k_can_go_up[k1] =1;
                                }
                                if(increase_possible >=0)
                                {
                                        v_k_for_e[k1] =       v_k_for_e[k1] +
increase_possible;
                                }
                                increase_possible=0;
                        }
                        objective_function_value_for_e =0;

                        for (k1 =1; k1 <=k; k1 ++)

                        {

                                printf("\n v[%d] =%d, ",k1, v_k_for_e[k1]);
                                objective_function_value_for_e =
objective_function_value_for_e +v_k_for_e[k1];

                        }

                        printf("\nObj. Fun Value =%d ",objective_function_value_for_e);

                        printf("give some number");

                        scanf("%d", &counter);
                        dual_ascent_can_go_on =1;
                        for (k1 =1; k1 <=k; k1 ++)
                                if (v_k_can_go_up[k1] ==0)
                                        dual_ascent_can_go_on =0;
                }
                objective_function_value_for_e =0;
                for (k1 =1; k1 <=k; k1 ++)
                        {
                                printf("\nv_k_for_e[%d]=%d, ",k1, v_k_for_e[k1]);
                                objective_function_value_for_e =
objective_function_value_for_e +v_k_for_e[k1];
                        }
                for (i1 =1; i1 <=i; i1 ++)
                        printf("\nRem_Resources[%d]=%d, ",i1,
remaining_resources_for_e[i1]);
                printf("\n Final Value For Objective Function Is: %d",
objective_function_value_for_e);
}
```

APPENDIX (B)

Table 1: Problem Set 10x10

Dual Ascent	SRS	Percentage Diff.	Ratio	No. Of Plants
f(i) =0				
9798	9798	0.00%	0.00%	Y:5
10222	10222	0.00%	0.00%	Y:6
11656	11656	0.00%	0.00%	Y:5
f(i) =1000				
14255	14255	0.00%	28.06%	Y:4
14832	14720	0.76%	20.23%	Y:3
16001	16001	0.00%	25.00%	Y:4
f(i) =2000				
18226	18226	0.00%	32.92%	Y:3
17051	17051	0.00%	23.46%	Y:2
19971	19818	0.77%	30.04%	Y:4
f(i) =3000				
21226	21226	0.00%	42.40%	Y:3
19051	19051	0.00%	31.49%	Y:2
23031	22305	3.25%	32.56%	Y:5
f(i) =5000				
26367.5	26028	1.30%	37.93%	Y:4
22574.5	22187	1.75%	33.22%	Y:3
27395	26515	3.32%	30.42%	Y:4
f(i) =10000				
32948	31568	4.37%	30.35%	Y:1
30063	29807	0.86%	33.26%	Y:1
35261	33857	4.15%	42.54%	Y:3
f(i) =20000				
42948	42948	0.00%	46.57%	Y:1
40063	40063	0.00%	49.92%	Y:1
48273	46793	3.16%	41.43%	Y:1

f(i) =25000				
47948	47948	0.00%	52.14%	Y:1
45063	45063	0.00%	55.48%	Y:1
53273	52755	0.98%	46.93%	Y:1

f(i) =30000				
52948	52948	0.00%	56.66%	Y:1
50063	50063	0.00%	59.92%	Y:1
58273	58273	0.00%	51.48%	Y:1

f(i) =50000				
72948	72948	0.00%	68.54%	Y:1
70063	70063	0.00%	71.36%	Y:1
78273	78273	0.00%	63.88%	Y:1

f(i) =100000				
122948	122948	0.00%	81.34%	Y:1
120063	120063	0.00%	83.29%	Y:1
128273	128273	0.00%	77.96%	Y:1

f(i) =200000				
222948	222948	0.00%	89.71%	Y:1
220063	220063	0.00%	90.88%	Y:1
228273	228273	0.00%	87.61%	Y:1

f(i) =500000				
522948	522948	0.00%	95.61%	Y:1
520063	520063	0.00%	96.14%	Y:1
528273	528273	0.00%	94.65%	Y:1

f(i) =1000000				
1022948	1022948	0.00%	97.76%	Y:1
1020063	1020063	0.00%	98.03%	Y:1
1028273	1028273	0.00%	97.25%	Y:1

f(i) =5000000				
5022948	5022948	0.00%	99.54%	Y:1
5020063	5020063	0.00%	99.60%	Y:1
5028273	5028273	0.00%	99.44%	Y:1

f(i) =8000000				
8022948	8022948	0.00%	99.71%	Y:1
8020063	8020063	0.00%	99.75%	Y:1
8028273	8028273	0.00%	99.65%	Y:1

f(i) =10000000				
10022948	10022948	0.00%	99.77%	Y:1
10020063	10020063	0.00%	99.80%	Y:1
10028273	10028273	0.00%	99.72%	Y:1

Table 2: Problem Set 20x20

Dual Ascent	SRS	Percentage Diff.	Ratio	No. Of Plants
f(i) =0				
15532	15532	0.00%	0.00%	Y:11
15654	15654	0.00%	0.00%	Y:11
15827	15827	0.00%	0.00%	Y:12
f(i) =1000				
23578	23422	0.67%	29.69%	Y:7
23238	23238	0.00%	21.52%	Y:5
22362	22017	1.57%	22.36%	Y:5
f(i) =2000				
28253	28105	0.53%	28.32%	Y:4
28238	27856	1.37%	35.41%	Y:5
27064.3	26503	2.12%	32.02%	Y:7
f(i) =3000				
32253	31957	0.93%	37.21%	Y:4
31990	31533	1.45%	28.13%	Y:3
31094	30698	1.29%	38.59%	Y:6
f(i) =5000				
38791	37502	3.44%	38.67%	Y:3
37990	37260	1.96%	39.48%	Y:3
37577	37327	0.67%	39.92%	Y:3
f(i) =8000				
45663	44508	2.60%	35.04%	Y:2
44928	41963	7.07%	29.68%	Y:4
45918.5	45620	0.65%	43.56%	Y:5
f(i) =10000				
49663	48824	1.72%	40.27%	Y:2
48071	44781	7.35%	31.20%	Y:3
49922	49562	0.73%	40.06%	Y:4

f(i) =20000				
63562	63562	0.00%	31.47%	Y:1
58247	58247	0.00%	34.34%	Y:1
64013	63842	0.27%	31.24%	Y:1

f(i) =25000				
68562	68562	0.00%	36.46%	Y:1
63247	63247	0.00%	39.53%	Y:1
69013	69013	0.00%	36.23%	Y:1

f(i) =30000				
73562	73562	0.00%	40.78%	Y:1
68247	68247	0.00%	43.96%	Y:1
74013	74013	0.00%	40.53%	Y:1

f(i) =50000				
93562	93562	0.00%	53.44%	Y:1
88247	88247	0.00%	56.66%	Y:1
94013	94013	0.00%	53.18%	Y:1

f(i) =100000				
143562	143562	0.00%	69.66%	Y:1
138247	138247	0.00%	72.33%	Y:1
144013	144013	0.00%	69.44%	Y:1

f(i) =200000				
243562	243562	0.00%	82.11%	Y:1
238247	238247	0.00%	83.95%	Y:1
244013	244013	0.00%	81.96%	Y:1

f(i) =500000				
543562	543562	0.00%	91.99%	Y:1
538247	538247	0.00%	92.89%	Y:1
544013	544013	0.00%	91.91%	Y:1

f(i) =1000000				
1043562	1043562	0.00%	95.83%	Y:1
1038247	1038247	0.00%	96.32%	Y:1
1044013	1044013	0.00%	95.78%	Y:1

f(i) =5000000				
5043562	5043562	0.00%	99.14%	Y:1
5038247	5038247	0.00%	99.24%	Y:1
5044013	5044013	0.00%	99.13%	Y:1

f(i) =8000000				
8043562	8043562	0.00%	99.46%	Y:1
8038247	8038247	0.00%	99.52%	Y:1
8044013	8044013	0.00%	99.45%	Y:1

f(i) =10000000				
10043562	10043562	0.00%	99.57%	Y:1
10038247	10038247	0.00%	99.62%	Y:1
10044013	10044013	0.00%	99.56%	Y:1

Table 3: Problem Set 30x30

Dual Ascent	SRS	Percentage Diff.	Ratio	No. Of Plants
21386	21386	0.00%	0.00%	Y:22
20222	20222	0.00%	0.00%	Y:18
26452	26452	0.00%	0.00%	Y:17
		f(i) =1000		
34261	34044	0.64%	26.27%	Y:9
29144	28933	0.73%	20.59%	Y:6
37570	37090	1.29%	21.29%	Y:8
		f(i) =2000		
42030.6	41661	0.89%	28.55%	Y:14
35144	34593	1.59%	34.15%	Y::6
44096	43770	0.74%	22.68%	Y:5
		f(i) =3000		
47304	46716	1.26%	31.71%	Y:11
39839.5	39027	2.08%	30.12%	Y:7
49096	48736	0.74%	30.55%	Y:5
		f(i) =5000		
55445.6	53382	3.87%	32.36%	Y:12
47723.5	46706	2.18%	39.29%	Y:11
58070.4	56231	3.27%	28.29%	Y:10
		f(i) =8000		
64786.3	61571	5.22%	35.50%	Y:11
57052.5	54968	3.79%	38.06%	Y:10
68278.1	66159	3.20%	32.95%	Y:12
		f(i) =10000		
62215	60192.000	3.36%	38.46%	Y:9
73318.8	69906.000	4.88%	32.73%	Y:8
		f(i) =20000		
89794	86577.000	3.72%	39.31%	Y:9
81135.3	78406.000	3.48%	41.08%	Y:4
93644.3	91342.000	2.52%	36.61%	Y:7

f(i) =25000				
98203.7	95004.000	3.37%	40.73%	Y:8
88694.5	85341.000	3.93%	42.28%	Y:3
102034	99519.000	2.53%	39.20%	Y:7

f(i) =30000				
105742	102071.000	3.60%	42.56%	Y:3
95730	91488.000	4.64%	43.87%	Y:6
109991	106632.000	3.15%	40.91%	Y:7

f(i) =50000				
132527	129071.000	2.68%	47.16%	Y:4
122638	116323.000	5.43%	54.36%	Y:4
133441	131491.000	1.48%	37.47%	Y:1

f(i) =100000				
184710	180610.000	2.27%	54.14%	Y:1
176223	172871.000	1.94%	56.75%	Y:1
183441	183441.000	0.00%	54.51%	Y:1

f(i) =200000				
284710	280610.000	1.46%	70.25%	Y:1
276223	272871.000	1.23%	72.41%	Y:1
283441	283441.000	0.00%	70.56%	Y:1

f(i) =500000				
584710	580610.000	0.71%	85.51%	Y:1
576223	572871.000	0.59%	86.77%	Y:1
583441	583441.000	0.00%	85.70%	Y:1

f(i) =1000000				
1084710	1080610.000	0.38%	92.19%	Y:1
1076223	1072871.000	0.31%	92.92%	Y:1
1083441	1083441.000	0.00%	92.30%	Y:1

f(i) =5000000				
5084710	5080610.000	0.08%	98.33%	Y:1
5076223	5072871	0.07%	98.50%	Y:1
5083441	5083441.000	0.00%	98.36%	Y:1

f(i) =8000000				
8084710	8080610	0.05%	98.95%	Y:1
8076223	8072871	0.04%	99.06%	Y:1
8083441	8083441	0.00%	98.97%	Y:1

f(i) =10000000				
10084710	10080610	0.04%	99.16%	Y:1
10076223	10072871	0.03%	99.24%	Y:1
10083441	10083441	0.00%	99.17%	Y:1

Table 3: Problem Set 40x40

Dual Ascent	SRS	Percentage Diff.	Ratio	No. Of Plants
f(i) =0				
26691	26691	0.00%	0.00%	Y:23
27986	27986	0.00%	0.00%	Y:23
22599	22599	0.00%	0.00%	Y:30
f(i) =1000				
40354	39693	1.67%	22.30%	Y:13
41915	40832	2.65%	23.86%	Y:10
36714	36018	1.93%	25.88%	Y:12
f(i) =2000				
48014	46668	2.88%	27.08%	Y:12
50685.3	49365	2.67%	28.61%	Y:12
44299	42813	3.47%	24.83%	Y:11
f(i) =3000				
53914.5	52982	1.76%	29.91%	Y:16
57075.2	55254	3.30%	30.08%	Y:14
49236.9	48095	2.37%	28.34%	Y:13
f(i) =5000				
62867.3	61204	2.72%	30.07%	Y:15
66604.2	63819	4.36%	28.48%	Y:13
57147.8	55859	2.31%	29.39%	Y:13
f(i) =8000				
72863	70613	3.19%	36.47%	Y:9
78065.8	74253	5.13%	33.31%	Y:9
65801.3	63488	3.64%	31.60%	Y:10
f(i) =10000				
78749.1	76187	3.36%	34.63%	Y:10
84376	80094	5.35%	34.99%	Y:13
70792.6	67494	4.89%	34.15%	Y:13

f(i) =20000				
100543	97338	3.29%	36.03%	Y:11
108971	103320	5.47%	36.71%	Y:6
91703.3	87144	5.23%	39.26%	Y:9

f(i) =25000				
109327	106239	2.91%	39.20%	Y:6
118373	113921	3.91%	39.22%	Y:6
100531	97257	3.37%	40.52%	Y:12

f(i) =30000				
117531	114866	2.32%	40.11%	Y:6
127194	122063	4.20%	41.28%	Y:5
108389	104091	4.13%	42.27%	Y:11

f(i) =50000				
139057	139057	0.00%	37.88%	Y:5
158386	150879	4.98%	35.95%	Y:1
136032	131744	3.25%	45.95%	Y:5

f(i) =100000				
189057	189057	0.00%	52.89%	Y:1
221366	209658	5.58%	45.17%	Y:5
196576	185779	5.81%	59.35%	Y:7

f(i) =200000				
289057	289057	0.00%	69.19%	Y:1
324809	309658	4.89%	61.57%	Y:1
299637	285779	4.85%	66.75%	Y:1

f(i) =500000				
589057	589057	0.00%	84.88%	Y:1
624809	609658	2.49%	80.02%	Y:1
599637	585779	2.37%	83.38%	Y:1

f(i) =1000000				
1089057	1089057	0.00%	91.82%	Y:1
1124809	1109658	1.37%	88.90%	Y:1
1099640	1085779	1.28%	90.94%	Y:1

f(i) =5000000				
5089057	5089057	0.00%	98.25%	Y:1
5124809	5109658	0.30%	97.56%	Y:1
5099640	5085779	0.27%	98.05%	Y:1

f(i) =8000000				
8089057	8089057	0.00%	98.90%	Y:1
8124809	8109658	0.19%	98.46%	Y:1
8099640	8095779	0.05%	98.77%	Y:1

f(i) =10000000				
10089057	10089057	0.00%	99.12%	Y:1
10124810	10109658	0.15%	98.77%	Y:1
10099640	10095779	0.04%	99.01%	Y:1

Table 5: Problem set 50x50

Dual Ascent	SRS	Percentage Diff.	Ratio	No. Of Plants
f(i) =0				
26691	26691	0.00%	0.00%	Y:23
27986	27986	0.00%	0.00%	Y:23
22599	22599	0.00%	0.00%	Y:30
f(i) =1000				
40354	39693	1.67%	22.30%	Y:13
41915	40832	2.65%	23.86%	Y:10
36714	36018	1.93%	25.88%	Y:12
f(i) =2000				
48014	46668	2.88%	27.08%	Y:12
50685.3	49365	2.67%	28.61%	Y:12
44299	42813	3.47%	24.83%	Y:11
f(i) =3000				
53914.5	52982	1.76%	29.91%	Y:16
57075.2	55254	3.30%	30.08%	Y:14
49236.9	48095	2.37%	28.34%	Y:13
f(i) =5000				
62867.3	61204	2.72%	30.07%	Y:15
66604.2	63819	4.36%	28.48%	Y:13
57147.8	55859	2.31%	29.39%	Y:13
f(i) =8000				
72863	70613	3.19%	36.47%	Y:9
78065.8	74253	5.13%	33.31%	Y:9
65801.3	63488	3.64%	31.60%	Y:10
f(i) =10000				
78749.1	76187	3.36%	34.63%	Y:10
84376	80094	5.35%	34.99%	Y:13
70792.6	67494	4.89%	34.15%	Y:13

f(i) =20000				
100543	97338	3.29%	36.03%	Y:11
108971	103320	5.47%	36.71%	Y:6
91703.3	87144	5.23%	39.26%	Y:9

f(i) =25000				
109327	106239	2.91%	39.20%	Y:6
118373	113921	3.91%	39.22%	Y:6
100531	97257	3.37%	40.52%	Y:12

f(i) =30000				
117531	114866	2.32%	40.11%	Y:6
127194	122063	4.20%	41.28%	Y:5
108389	104091	4.13%	42.27%	Y:11

f(i) =50000				
139057	139057	0.00%	37.88%	Y:5
158386	150879	4.98%	35.95%	Y:1
136032	131744	3.25%	45.95%	Y:5

f(i) =100000				
189057	189057	0.00%	52.89%	Y:1
221366	209658	5.58%	45.17%	Y:5
196576	185779	5.81%	59.35%	Y:7

f(i) =200000				
289057	289057	0.00%	69.19%	Y:1
324809	309658	4.89%	61.57%	Y:1
299637	285779	4.85%	66.75%	Y:1

f(i) =500000				
589057	589057	0.00%	84.88%	Y:1
624809	609658	2.49%	80.02%	Y:1
599637	585779	2.37%	83.38%	Y:1

f(i) =1000000				
1089057	1089057	0.00%	91.82%	Y:1
1124809	1109658	1.37%	88.90%	Y:1
1099640	1085779	1.28%	90.94%	Y:1

f(i) =5000000				
5089057	5089057	0.00%	98.25%	Y:1
5124809	5109658	0.30%	97.56%	Y:1
5099640	5085779	0.27%	98.05%	Y:1

f(i) =8000000				
8089057	8089057	0.00%	98.90%	Y:1
8124809	8109658	0.19%	98.46%	Y:1
8099640	8095779	0.05%	98.77%	Y:1

f(i) =10000000				
10089057	10089057	0.00%	99.12%	Y:1
10124810	10109658	0.15%	98.77%	Y:1
10099640	10095779	0.04%	99.01%	Y:1

Table 6: Problem set 60x60

Dual Ascent	SRS	Percentage Diff.	Ratio	No. Of Plants
f(i) =0				
36515	36515	0.00%	0.00%	Y:34
34352	34352	0.00%	0.00%	Y:35
35729	35729	0.00%	0.00%	Y:33
f(i) =1000				
55748	54836	1.66%	26.91%	Y:15
54214.5	53295	1.73%	22.60%	Y:22
53049	51702	2.61%	21.68%	Y:22
f(i) =2000				
66655.2	65155	2.30%	26.88%	Y:25
64092	62318	2.85%	26.52%	Y:11
62250.3	59995	3.76%	26.02%	Y:20
f(i) =3000				
74268	73045	1.67%	27.15%	Y:25
72078	69338	3.95%	30.35%	Y:18
69830.8	67820	2.96%	29.54%	Y:21
f(i) =5000				
85896.9	84105	2.13%	28.69%	Y:24
84290.4	81853	2.98%	31.98%	Y:19
81926.3	79973	2.44%	33.39%	Y:19
f(i) =8000				
99396.6	96177	3.35%	33.09%	Y:19
99085.1	96032	3.18%	35.50%	Y:18
96861.9	93925	3.13%	37.08%	Y:22
f(i) =10000				
107181	103204	3.85%	34.20%	Y:23
107262	104513	2.63%	35.76%	Y:15
104998	101575	3.37%	35.71%	Y:11

f(i) =20000				
135838	131134	3.59%	34.74%	Y:18
138345.6	134080	3.18%	38.04%	Y:14
135003.3	128684	4.91%	36.02%	Y:15

f(i) =25000				
147076	141931	3.63%	36.68%	Y;16
150413	145765	3.19%	38.23%	Y:13
146547	139930	4.73%	37.18%	Y:14

f(i) =30000				
157369	151752	3.70%	38.99%	Y:15
161287	156577	3.01%	38.34%	Y:14
156951	150684	4.16%	38.46%	Y:14

f(i) =50000				
193059	186781	3.36%	40.29%	Y:7
196985	188979	4.24%	39.73%	Y:11
192681	185872	3.66%	38.92%	Y:5

f(i) =100000				
260807	253712	2.80%	47.93%	Y:5
264338	254632	3.81%	47.29%	Y:5
259658	253291	2.51%	46.21%	Y:6

f(i) =200000				
374919	355534	5.45%	59.27%	Y:10
371782	358172	3.80%	53.79%	Y:1
374474	357706	4.69%	59.34%	Y:10

f(i) =500000				
699000	655534	6.63%	71.53%	Y:1
671782	658172	2.07%	74.43%	Y:1
684372	657706	4.05%	77.93%	Y:16

f(i) =1000000				
1232258	1155534	0.066397008	86.33%	Y:22
1171782	1158172	0.011751277	85.34%	Y:1
1184372	1157706	0.023033482	84.43%	Y:1

f(i) =5000000				
5259739	5155534	0.020212261	95.06%	Y:1
5171782	5158172	0.002638532	96.68%	Y:1
5184372	5157706	0.005170128	96.44%	Y:1

f(i) =8000000				
8670420	8155534	0.063133328	92.27%	Y:1
8171782	8158172	0.001668266	97.90%	Y:1
8184372	8158172	0.003211504	97.75%	Y:1

f(i) =10000000				
10795420	1E+07	6.30%	98.42%	Y:17
10171780	1E+07	0.13%	98.31%	Y:1
10184373	1E+07	0.26%	98.19%	Y:1

Table 1: Problem Set 10x10

Dual Ascent	SRS	Percentage Diff.	Ratio	No. Of Plants
f(i) =0				
31666	31666	0.00%	0.00%	Y:43
40161	40161	0.00%	0.00%	Y:41
41127	41127	0.00%	0.00%	Y:42
f(i) =1000				
51350	49964	2.77%	22.20%	Y:23
60862.8	59490	2.31%	24.32%	Y:25
63856.5	62090	2.85%	21.92%	Y:25
f(i) =2000				
61493.6	59455	3.43%	29.42%	Y:28
72262.8	70432	2.60%	26.45%	Y:29
75307.5	73120	2.99%	26.56%	Y:19
f(i) =3000				
69482.9	67729	2.59%	31.25%	Y:24
81007.2	78284	3.48%	29.61%	Y:30
84131.3	82069	2.51%	27.12%	Y:24
f(i) =5000				
81583.1	79049	3.21%	32.34%	Y:27
94257.2	91468	3.05%	30.33%	Y:26
97269.7	94828	2.57%	29.59%	Y:26
f(i) =8000				
95410.3	92946	2.65%	35.71%	Y:25
109294	104942	4.15%	32.94%	Y:18
112508	107677	4.49%	31.72%	Y:16
f(i) =10000				
103577	101380	2.17%	37.74%	Y:26
117910	113489	3.90%	35.54%	Y:25
120818	115514	4.59%	33.35%	Y:17

f(i) =25000				
146295	141710	3.24%	37.91%	Y:15
163928	159507	2.77%	37.76%	Y:17
166659	155848	6.94%	37.13%	Y:17

f(i) =50000				
191390	185317	3.28%	39.19%	Y:8
215072	210986	1.94%	39.85%	Y:12
218380	205351	6.34%	41%	Y:13

f(i) =100000				
255901	248722	2.89%	45.60%	Y:7
286935	280217	2.40%	47.29%	Y:8
293295	279693	4.86%	42.62%	Y:8

f(i) =150000				
313244.1	300084	4.39%	53.20%	Y:12
346661	340427	1.83%	53.20%	Y:8
355145.8	337450	5.24%	53.20%	Y:6

f(i) =200000				
368738.6	350084	5.33%	59.57%	Y:17
403802.4	390427	3.43%	59.57%	Y:8
414436.3	387450	6.97%	59.57%	Y:8

f(i) =500000				
677335	650084	4.19%	73.82%	Y:1
720876	690427	4.41%	69.36%	Y:1
723670	687450	5.27%	69.09%	Y:1

f(i) =1000000				
1177335	1E+06	2.37%	84.30%	Y:1
1220876	1E+06	2.56%	84.30%	Y:1
1223670	1E+06	3.05%	84.30%	Y:1

f(i) =2500000				
2677335	3E+06	1.03%	93.38%	Y:1
2720876	3E+06	1.13%	91.88%	Y:1
2723670	3E+06	1.35%	91.79%	Y:1

f(i) =5000000				
5177335	5E+06	0.53%	96.57%	Y:1
5220876	5E+06	0.59%	95.77%	Y:1
5223670	5E+06	0.70%	95.72%	Y:1

f(i) =8000000				
8177335	8E+06	0.33%	97.83%	Y:1
8220876	8E+06	0.37%	97.31%	Y:1
8223670	8E+06	0.44%	97.28%	Y:1

f(i) =10000000				
10177340	1E+07	0.27%	98.26%	Y:1
10220880	1E+07	0.30%	97.84%	Y:1
10223670	1E+07	0.36%	97.81%	Y:1

APPENDIX (C)

Lingo formulation for Strong Relaxation for SPLP(SRS)

```
!FORMULATION IS FOR OBJECTIVE VALUE WITH RELAXATION;
!S IS THE CAPACITY IN FRACTION I.E. /TOTAL DEMAND;
!D IS DEMAND AT EACH POINT IN FRACTION;
!C IS COST OF TTRANSPORTING TOTAL DEMAND OF K FROM POINT I;
!X IS FRACTION OF TRANSPORTATION FROM I TO K WITH RESOPECT TO DEMAND AT
K;
!F IS THE FIXED COST INCURRED FOR ESTABLISHING PLANT AT LOCATION I;
!Y IS A VARIABLE EQUALS 1 WHEN PLANT IS INCLUDED AND 0 OTHERWISE;
SETS:
       SUPPLY /1..20/: F, Y;
       DEMAND /1..20/: D;
       TRANSPORT (SUPPLY, DEMAND): C, X;
ENDSETS
DATA:


ENDDATA
       !OBJECTIVE FUNCTION;
MIN = @SUM(TRANSPORT(I, K):C(I, K)*X(I, K))+@SUM(SUPPLY(I):F(I)*Y(I));


!SUBJECT TO;
       !SUM OVER I Xik = 1 FOR ALL K;
@FOR(DEMAND(K):@SUM(SUPPLY(I):X(I, K))=1);
       ! Strong Relaxation;
       !Yi-Xik >= 0 FOR ALL I AND K;
@FOR(TRANSPORT(I, K): Y(I)-X(I, K)>=0);
```

Constraints added for Big M formulation:

```
!BIG 'M' CONSTRAINT;
       !SUM OVER K OF Xik + M(1-Yi) >= 0 FOR ALL I;
@FOR(SUPPLY(I):(@SUM(DEMAND(K):X(I, K))+M*(1-Y(I)))>=0);
       !SUM OVER K OF Xik - M*Yi <= 0 FOR ALL I;
@FOR(SUPPLY(I):(@SUM(DEMAND(K):X(I, K))-M*Y(I))<=0);
       !SUM OVER K OF Xik + M*Yi >= 0 FOR ALL I;
@FOR(SUPPLY(I):(@SUM(DEMAND(K):X(I, K))+M*Y(I))>=0);
```

Lingo formulation for Weak Relaxation for SPLP(New)

```
SETS:
      SUPPLY /1..20/: F, Y;
      DEMAND /1..20/: D;
      TRANSPORT (SUPPLY, DEMAND): C, X;
ENDSETS
DATA:


ENDDATA
      !OBJECTIVE FUNCTION;
MIN = @SUM(TRANSPORT(I, K):C(I, K)*X(I, K))+@SUM(SUPPLY(I):F(I)*Y(I));

!SUBJECT TO;
      !SUM OVER I AND K Xik = 1;
@SUM(TRANSPORT(I, K):X(I, K))=1;
!Weak Relaxation;
      !(Yi-SUM OVER K Xik)>=0 FOR ALL I;
@FOR(SUPPLY(I):(Y(I)-@SUM(D(K):X(I, K)))>=0);
      !(Dk-SUM OVER i Xik)>=0 FOR ALL K;
@FOR(DEMAND(K):(D(K)-@SUM(SUPPLY(I):X(I, K)))>=0);
```

Constraints added for Big M formulation

```
!BIG 'M' CONSTRAINT;
      !SUM OVER K OF Xik + M(1-Yi) >= 0 FOR ALL I;
@FOR(SUPPLY(I):(@SUM(DEMAND(K):X(I, K))+M*(1-Y(I)))>=0);
      !SUM OVER K OF Xik - M*Yi <= 0 FOR ALL I;
@FOR(SUPPLY(I):(@SUM(DEMAND(K):X(I, K))-M*Y(I))<=0);
      !SUM OVER K OF Xik + M*Yi >= 0 FOR ALL I;
@FOR(SUPPLY(I):(@SUM(DEMAND(K):X(I, K))+M*Y(I))>=0);
```

Lingo formulation for Weak Relaxation for SPLP(Old)

```
SETS:
        SUPPLY /1..20/: F, Y;
        DEMAND /1..20/: D;
        TRANSPORT (SUPPLY, DEMAND): C, X;
ENDSETS
DATA:


ENDDATA
        !OBJECTIVE FUNCTION;
MIN = @SUM(TRANSPORT(I, K):C(I, K)*X(I, K))+@SUM(SUPPLY(I):F(I)*Y(I));

!SUBJECT TO;
        !SUM OVER I AND K Xik = 1;
@SUM(TRANSPORT(I, K):X(I, K))=1;
!Weak Relaxation;
        !(Yi-(1/K)SUM OVER K Xik/Dk)>=0 FOR ALL I;
@FOR(SUPPLY(I):(Y(I)-(1/K1)*@SUM(DEMAND(K):X(I, K)/DMD(K)))>=0);
        !(Dk-SUM OVER i Xik)>=0 FOR ALL K;
@FOR(DEMAND(K):(DMD(K)-@SUM(SUPPLY(I):X(I, K)))>=0);
```

Constraints added for Big M formulation

```
!BIG 'M' CONSTRAINT;
        !SUM OVER K OF Xik + M(1-Yi) >= 0 FOR ALL I;
@FOR(SUPPLY(I):(@SUM(DEMAND(K):X(I, K))+M*(1-Y(I)))>=0);
        !SUM OVER K OF Xik - M*Yi <= 0 FOR ALL I;
@FOR(SUPPLY(I):(@SUM(DEMAND(K):X(I, K))-M*Y(I))<=0);
        !SUM OVER K OF Xik + M*Yi >= 0 FOR ALL I;
@FOR(SUPPLY(I):(@SUM(DEMAND(K):X(I, K))+M*Y(I))>=0);
```

APPENDIX (D)

Problems 10x10

Problem No.	*First Formulation*	*Second Formulation*	*Dual Ascent*	*SRS*
	Value	*Value*	*Value*	*Value*
1	11727.52	11819.60	16819.00	16819.00
2	12146.35	12129.80	17133.00	17133.00
3	12757.70	13003.40	16251.00	16251.00
4	14262.24	14577.00	18436.00	18601.00
5	11893.18	11915.60	17458.00	17458.00
6	9836.09	9798.70	15303.00	15303.00
7	12355.00	12205.70	16568.00	16756.00
8	13785.35	13595.00	18497.00	18497.00
9	10450.66	10615.00	15321.00	15321.00
10	12173.24	12105.80	18794.00	19041.00
11	15148.50	15229.80	19130.00	19536.00
12	17740.99	17586.20	22067.00	22095.00
13	8984.76	8820.80	14547.00	14791.00
14	10516.15	10257.20	14602.00	14602.00
15	16244.03	16214.30	21070.00	21070.00
16	12182.61	12226.00	17180.00	17180.00
17	11288.90	11522.00	15163.00	15163.00
18	16160.06	15917.10	25307.00	25307.00
19	9801.01	9265.80	13656.00	14252.30
20	12956.30	12707.50	17515.00	17515.00
21	14073.57	14083.40	17736.00	17736.00
22	12352.37	12275.00	17067.00	17067.00
23	15945.21	15771.50	20183.00	20183.00
24	16229.07	16064.40	22550.00	22691.00
25	10278.42	10196.20	14156.00	14156.00
26	12834.77	12803.30	18722.00	19149.00
27	13722.08	13516.60	20293.00	20239.00
28	10367.89	10256.50	17620.00	18195.00
29	15305.92	15349.80	23449.00	23449.00
30	15281.52	15271.50	19373.00	19373.00

Problems 20x20

Problem No.	First Formulation	Second Formulation	Dual Ascent	SRS
	Value	Value	Value	Value
1	17604.44	17454.60	26546	27703
2	17806.77	17931.50	29246	29438
3	18330.35	18278.65	29767	30238
4	19762.42	19761.35	29268	29924
5	14864.63	14623.70	22095	22606
6	19427.93	19457.35	29254	29534
7	14796.81	14816.95	23292	23292
8	16091.94	16055.90	23138	23214
9	17492.78	17359.45	30999	31776.3
10	18803.35	18805.30	29063	29262
11	14300.19	14010.65	23889	24354
12	15968.32	16043.60	28981	29072
13	15730.04	15545.55	27798	27996
14	19228.37	19410.25	26433	26521
15	19766.58	19638.55	30973	31258
16	17147.28	17005.60	25254	25644
17	18546.50	18577.10	27907	28240
18	15099.29	15270.55	22443	22623
19	21116.65	21176.20	32931	32931
20	20150.16	20168.50	33837	33837
21	13853.63	13660.90	19899	20378
22	21452.23	21418.45	31565	32012
23	19758.10	19601.45	31949	32401.7
24	15447.75	15378.75	25602	25602
25	19592.89	19540.05	31129	31542
26	14555.67	14595.85	25325	26053.5
27	19716.43	19721.85	31681	32005
28	21980.85	21951.10	33031	33636
29	16693.19	16650.30	26201	26478
30	16310.91	16361.60	24607	25121.5

Problems 30x30

Problem No.	First Formulation	Second Formulation	Dual Ascent	SRS
	Value	Value	Value	Value
1	23682.57	23557.87	41043.00	41244.00
2	22196.37	22013.10	31349.00	31891.00
3	28756.03	28751.03	43759.00	44194.00
4	24969.36	24909.43	41702.00	42682.00
5	24793.61	24865.77	39745.00	40737.00
6	22541.06	22512.50	38476.00	38709.50
7	22884.45	22882.43	39636.00	40980.00
8	27066.34	26875.56	42084.00	42947.00
9	25372.30	25279.96	41106.00	41987.00
10	23935.23	23815.16	34322.00	34322.00
11	22882.39	22837.24	35395.00	36061.00
12	26607.57	26610.00	41212.00	41576.00
13	23822.19	23769.67	34512.00	34962.00
14	28946.14	28894.57	42378.00	42579.00
15	29483.48	29597.46	45472.00	45575.00

Problems 40x40

Problem No.	First Formulation	Second Formulation	Dual Ascent	SRS
	Value	Value	Value	Value
1	28812.05	28791.59	44850.00	45414.00
2	30153.78	30110.58	48534.00	49727.00
3	24592.53	24542.33	41303.00	42422.00
4	25475.09	25553.30	39012.00	39572.50
5	25119.50	25158.45	42085.00	42477.00
6	30392.07	30383.39	51217.00	52275.50
7	29794.22	29767.05	48463.00	49348.00
8	25465.47	25522.60	41176.00	41879.00
9	25596.03	25440.95	42795.00	43879.50
10	25852.32	25714.64	46367.00	46761.50
11	23608.43	23727.50	38765.00	39137.00
12	28129.06	28174.71	41894.00	43740.50
13	29540.87	29542.66	47190.00	47996.70
14	26973.88	26854.15	44461.00	46031.00
15	28908.45	28794.35	48441.00	50333.00

Problems 50x50

Problem No.	First Formulation	Second Formulation	Dual Ascent	SRS
	Value	Value	Value	Value
1	29506.85	29351.26	47129.00	47755.00
2	35762.38	35692.99	57059.00	57433.00
3	26834.96	26721.97	44660.00	46373.00
4	31689.63	31479.72	54604.00	55773.00
5	32338.47	32349.86	50175.00	51064.00
6	30995.14	30981.89	48393.00	49688.00
7	29175.98	29150.92	47884.00	48672.00
8	30613.26	30525.67	49168.00	49767.00
9	30247.01	30157.08	46949.00	47488.00
10	28886.02	28872.46	45942.00	47798.00
11	30270.80	30135.31	48525.00	49891.00
12	27620.05	27486.05	46015.00	47385.00
13	27168.85	21027.68	45398.00	47223.00
14	25347.96	25438.70	42758.00	42978.00
15	28653.88	28546.26	47688.00	48650.00

Problems 60x60

Problem No.	First Formulation	second Formulation	Dual Ascent	SRS
	Value	Value	Value	Value
1	38438.63	38330.29	61870.00	62874.00
2	36321.41	36228.31	61879.00	63185.00
3	38183.43	38085.93	60105.00	61279.70
4	35124.81	35121.00	57737.00	59137.50
5	36223.79	36169.59	54202.00	55110.00
6	36126.27	36113.87	55737.00	57388.00
7	37199.36	37266.85	59112.00	60157.00
8	35964.68	35876.88	59712.00	60965.00
9	38324.57	38258.85	60241.00	61464.00
10	32297.06	32131.34	54798.00	55549.00
11	34139.79	34085.93	53078.00	53897.00
12	34591.05	34635.13	54772.00	55913.00
13	37365.86	37228.39	60220.00	62189.30
14	39405.84	39417.15	60358.00	61301.00
15	30846.70	30876.45	52429.00	52977.00

Problem No.	First Formulation	Second Formulation	Dual Ascent	SRS
	Value	Value	Value	Value
1	29506.85	29351.26	47129.00	47755.00
2	35762.38	35692.99	57059.00	57433.00
3	26834.96	26721.97	44660.00	46373.00
4	31689.63	31479.72	54604.00	55773.00
5	32338.47	32349.86	50175.00	51064.00
6	30995.14	30981.89	48393.00	49688.00
7	29175.98	29150.92	47884.00	48672.00
8	30613.26	30525.67	49168.00	49767.00
9	30247.01	30157.08	46949.00	47488.00
10	28886.02	28872.46	45942.00	47798.00
11	30270.80	30135.31	48525.00	49891.00
12	27620.05	27486.05	46015.00	47385.00
13	27168.85	21027.68	45398.00	47223.00
14	25347.96	25438.70	42758.00	42978.00
15	28653.88	28546.26	47688.00	48650.00

Problems 60x60

Problem No.	First Formulation	second Formulation	Dual Ascent	SRS
	Value	Value	Value	Value
1	38438.63	38330.29	61870.00	62874.00
2	36321.41	36228.31	61879.00	63185.00
3	38183.43	38085.93	60105.00	61279.70
4	35124.81	35121.00	57737.00	59137.50
5	36223.79	36169.59	54202.00	55110.00
6	36126.27	36113.87	55737.00	57388.00
7	37199.36	37266.85	59112.00	60157.00
8	35964.68	35876.88	59712.00	60965.00
9	38324.57	38258.85	60241.00	61464.00
10	32297.06	32131.34	54798.00	55549.00
11	34139.79	34085.93	53078.00	53897.00
12	34591.05	34635.13	54772.00	55913.00
13	37365.86	37228.39	60220.00	62189.30
14	39405.84	39417.15	60358.00	61301.00
15	30846.70	30876.45	52429.00	52977.00

Problems 70x70

Problem No.	*First Formulation*	*Second Formulation*	*Dual Ascent*	*SRS*
	Value	*Value*	*Value*	*Value*
1	33627.40	33459.86	55027.00	55699.00
2	42113.36	42145.36	65352.00	67022.50
3	42986.06	42986.80	66473.00	67687.00
4	44550.39	44504.91	72129.00	73906.30
5	38662.61	38469.26	58756.00	60704.00
6	38046.95	37848.06	61395.00	62165.00
7	38221.22	38212.69	63994.00	64940.00
8	42715.03	42742.62	66865.00	67908.00
9	44196.80	44132.65	68777.00	70353.50
10	38230.18	38276.02	67582.00	69066.50
11	41662.99	41747.68	60501.00	61585.00
12	41727.04	41642.59	69447.00	71307.00
13	39348.89	39265.56	69736.00	71239.30
14	39578.69	39385.48	67911.00	68691.70
15	39057.56	38856.25	63465.00	65768.70

APPENDIX (E)

OBJECTIVE VALUE OF CPLPL WITHOUT RELAXATION

```
!FORMULATION IS FOR OBJECTIVE VALUE WITHOUT RELAXATING ANY CONSTRAINTS;
!S IS THE CAPACITY IN FRACTION I.E. /TOTAL DEMAND;
!D IS DEMAND AT EACH POINT IN FRACTION;
!C IS COST OF TTRANSPORTING TOTAL DEMAND OF K FROM POINT I;
!X IS FRACTION OF TRANSPORTATION FROM I TO K WITH RESOPECT TO DEMAND AT K;
!F IS THE FIXED COST INCURRED FOR ESTABLISHING PLANT AT LOCATION I;
!Y IS A VARIABLE EQUALS 1 WHEN PLANT IS INCLUDED AND 0 OTHERWISE;

SETS:
      SUPPLY /1..20/: F, Y, S;
      DEMAND /1..20/: D, V ;
      TRANSPORT (SUPPLY, DEMAND): C, X;
ENDSETS

DATA:

ENDDATA
      !OBJECTIVE FUNCTION;
MIN = @SUM(TRANSPORT(I, K):C(I, K)*X(I, K))+@SUM(SUPPLY(I):F(I)*Y(I));

!SUBJECT TO;
      !SUM Xik = 1;
@SUM(TRANSPORT(I, K):X(I, K))=1;
      !SUM OVER I OF Xik IS <= Dk FOR ALL K;
@FOR(DEMAND(K):@SUM(S(I):X(I, K))<=D(K));
      !SUM OVER K OF Xik IS <= Si FOR ALL I;
@FOR(SUPPLY(I):@SUM(D(K):X(I, K))<=S(I));
      !Xik <= Yi*Dk FOR ALL I AND K;
@FOR(TRANSPORT(I, K):X(I, K)<=Y(I)*D(K));
      ! Y IS EITHER 0 OR 1;
@FOR(SUPPLY(I):@BIN(Y(I)));
```

SRS (RELAXATTION: P1)

```
!FORMULATION IS FOR OBJECTIVE VALUE WITHOUT RELAXATIoN;
!S IS THE CAPACITY IN FRACTION I.E. /TOTAL DEMAND;
!D IS DEMAND AT EACH POINT IN FRACTION;
!C IS COST OF TTRANSPORTING TOTAL DEMAND OF K FROM POINT I;
!X IS FRACTION OF TRANSPORTATION FROM I TO K WITH RESOPECT TO DEMAND AT K;
!F IS THE FIXED COST INCURRED FOR ESTABLISHING PLANT AT LOCATION I;
!Y IS A VARIABLE EQUALS 1 WHEN PLANT IS INCLUDED AND 0 OTHERWISE;

SETS:
      SUPPLY /1..20/: F, Y, S;
      DEMAND /1..20/: D, V ;
      TRANSPORT (SUPPLY, DEMAND): C, X;
ENDSETS

DATA:



ENDDATA

      !OBJECTIVE FUNCTION;
MIN = @SUM(TRANSPORT(I, K):C(I, K)*X(I, K))+@SUM(SUPPLY(I):F(I)*Y(I));

!SUBJECT TO;
      !SUM Xik = 1;
@SUM(TRANSPORT(I, K):X(I, K))=1;
      !SUM OVER I OF Xik IS <= Dk FOR ALL K;
@FOR(DEMAND(K):@SUM(S(I):X(I, K))<=D(K));
      !SUM OVER K OF Xik IS <= Si FOR ALL I;
@FOR(SUPPLY(I):@SUM(D(K):X(I, K))<=S(I));
      !Xik <= Yi*Dk FOR ALL I AND K;
@FOR(TRANSPORT(I, K):X(I, K)<=Y(I)*D(K));
```

Relaxation : P2

```
!FORMULATION IS FOR OBJECTIVE VALUE WITH RELAXATion;
!S IS THE CAPACITY IN FRACTION I.E. /TOTAL DEMAND;
!D IS DEMAND AT EACH POINT IN FRACTION;
!C IS COST OF TTRANSPORTING TOTAL DEMAND OF K FROM POINT I;
!X IS FRACTION OF TRANSPORTATION FROM I TO K WITH RESOPECT TO DEMAND AT K;
!F IS THE FIXED COST INCURRED FOR ESTABLISHING PLANT AT LOCATION I;
!Y IS A VARIABLE EQUALS 1 WHEN PLANT IS INCLUDED AND 0 OTHERWISE;

SETS:
        SUPPLY /1..20/: F, Y, S;
        DEMAND /1..20/: D, V ;
        TRANSPORT (SUPPLY, DEMAND): C, X;
ENDSETS


DATA:


ENDDATA
        !OBJECTIVE FUNCTION;
MIN = @SUM(TRANSPORT(I, K):C(I, K)*X(I, K))+@SUM(SUPPLY(I):F(I)*Y(I));


!SUBJECT TO;
        !SUM Xik = 1;
@SUM(TRANSPORT(I, K):X(I, K))=1;
        !SUM OVER I OF Xik IS <= Dk FOR ALL K;
@FOR(DEMAND(K):@SUM(S(I):X(I, K))<=D(K));
        !Xik IS <= Yi*Dk FOR ALL I AND K;
@FOR(TRANSPORT(I, K):X(I, K)<=Y(I)*DMD(K));
!BIG ‘M’ CONSTRAINT;
        !SUM OVER K OF Xik - M(1-Yi) <= Si FOR ALL I;
@FOR(SUPPLY(I):(@SUM(DEMAND(K):X(I, K))-M*(1-Y(I)))<=SUPPL(I));
        !SUM OVER K OF Xik - M*Yi <= 0 FOR ALL I;
@FOR(SUPPLY(I):(@SUM(DEMAND(K):X(I, K))-M*Y(I))<=0);
        !SUM OVER K OF Xik + M*Yi <= 0 FOR ALL I;
@FOR(SUPPLY(I):(@SUM(DEMAND(K):X(I, K))+M*Y(I)>=0);
```

RALXATION : P3

```
!FORMULATION IS FOR OBJECTIVE VALUE WITH RELAXATION;
!S IS THE CAPACITY IN FRACTION I.E. /TOTAL DEMAND;
!D IS DEMAND AT EACH POINT IN FRACTION;
!C IS COST OF TTRANSPORTING TOTAL DEMAND OF K FROM POINT I;
!X IS FRACTION OF TRANSPORTATION FROM I TO K WITH RESOPECT TO DEMAND AT K;
!F IS THE FIXED COST INCURRED FOR ESTABLISHING PLANT AT LOCATION I;
!Y IS A VARIABLE EQUALS 1 WHEN PLANT IS INCLUDED AND 0 OTHERWISE;

SETS:
      SUPPLY /1..20/: F, Y, S;
      DEMAND /1..20/: D, V ;
      TRANSPORT (SUPPLY, DEMAND): C, X;
ENDSETS

DATA:



ENDDATA
      !OBJECTIVE FUNCTION;
MIN = @SUM(TRANSPORT(I, K):C(I, K)*X(I, K))+@SUM(SUPPLY(I):F(I)*Y(I));

!SUBJECT TO;
      !SUM Xik = 1;
@SUM(TRANSPORT(I, K):X(I, K))=1;
      !SUM OVER I OF Xik IS <= Dk FOR ALL K;
@FOR(DEMAND(K):@SUM(S(I):X(I, K))<=D(K));
      !Xik <= Yi*Dk FOR ALL I AND K;
@FOR(TRANSPORT(I, K):X(I, K)<=Y(I)*D(K));
      !SUM OVER K OF Xik IS <= Si*Yi FOR ALL I;
@FOR(SUPPLY(I):@SUM(DEMAND(K):X(I, K))<= S(I)*Y(I));
```

Relaxation : P4

```
!FORMULATION IS FOR OBJECTIVE VALUE WITH RELAXATION;
!S IS THE CAPACITY IN FRACTION I.E. /TOTAL DEMAND;
!D IS DEMAND AT EACH POINT IN FRACTION;
!C IS COST OF TTRANSPORTING TOTAL DEMAND OF K FROM POINT I;
!X IS FRACTION OF TRANSPORTATION FROM I TO K WITH RESOPECT TO DEMAND
!F IS THE FIXED COST INCURRED FOR ESTABLISHING PLANT AT LOCATION I;
!Y IS A VARIABLE EQUALS 1 WHEN PLANT IS INCLUDED AND 0 OTHERWISE;

SETS:
        SUPPLY /1..20/: F, Y, S;
        DEMAND /1..20/: D, V ;
        TRANSPORT (SUPPLY, DEMAND): C, X;
ENDSETS


DATA:


ENDDATA
        !OBJECTIVE FUNCTION;
MIN = @SUM(TRANSPORT(I, K):C(I, K)*X(I, K))+@SUM(SUPPLY(I):F(I)*Y(I));


!SUBJECT TO;
        !SUM Xik = 1;
@SUM(TRANSPORT(I, K):X(I, K))=1;
        !SUM OVER I OF Xik IS <= Dk FOR ALL K;
@FOR(DEMAND(K):@SUM(S(I):X(I, K))<=D(K));
        !Xik <= Yi*Dk FOR ALL I AND K;
@FOR(TRANSPORT(I, K):X(I, K)<=Y(I)*D(K));
        !SUM OVER K OF Xik IS <= Si*Yi FOR ALL I;
@FOR(SUPPLY(I):@SUM(DEMAND(K):X(I, K))<= S(I)*Y(I));
!BIG 'M' CONSTRAINT;
        !SUM OVER K OF Xik - M(1-Yi) <= Si FOR ALL I;
@FOR(SUPPLY(I):(@SUM(DEMAND(K):X(I, K))-M*(1-Y(I)))<=SUPPL(I));
        !SUM OVER K OF Xik - M*Yi <= 0 FOR ALL I;
@FOR(SUPPLY(I):(@SUM(DEMAND(K):X(I, K))-M*Y(I))<=0);
        !SUM OVER K OF Xik + M*Yi >= 0 FOR ALL I;
@FOR(SUPPLY(I):(@SUM(DEMAND(K):X(I, K))+M*Y(I)>=0);
```

Table-1 Relaxation: P1(SRS) V/s Relaxation: P2

Problem No.	Relaxation :P1	Iterations	Relaxation: P2	Iterations
1	41814	1949	39884.5	3179
2	45832.9	1999	41540	1658
3	42665.5	2280	40274.7	1752
4	47368	1905	41384	1678
5	35669.9	2053	32650	3117
6	42208.8	1567	37673.5	2955
7	46735.1	1905	44121	1692
8	48929.8	1529	42176	1684
9	45182.4	2028	41235	1730
10	36467.1	1544	33635	1404
11	41685.2	1356	38287.3	1698
12	43404.1	1970	39268	1835
13	41563	1816	36811	1747
14	42198.7	1651	36226	1625
15	44578	1766	40347	2053
16	41352.9	1735	37726	1415
17	46578.8	1796	42262	1623
18	39547.3	1287	38115	2029
19	42643.3	1996	37510	1477
20	38335.8	1648	35260	1558
21	38868.5	1659	34849	3026
22	47660.1	1278	44828	1620
23	48448	2048	33609	1599
24	43629.3	1839	34939	3039
25	37420	1485	31152	2047
26	43281.6	2052	37300	1852
27	35540.6	2111	31832	3224
28	35571.6	1435	32839	1585
29	46888.3	1932	42879	1659
30	45915.1	1965	41429	1610
31	53826.6	2385	45261	1719
32	36114	2205	33883	1605
33	37520.1	1762	34323	1523
34	51172.7	2073	43314	1709
35	36724.26	1717	31447	1439
36	34846.7	1312	30809	1449
37	41304.7	1510	37650	1552
38	36149.7	1955	30638	3083
39	47329.2	1971	43891.3	1573
40	39667.1	2075	38126	1539
41	39841.3	1967	35979	1526
42	38522.13	1519	32492	1688

43	35492	2087	30763	2940
44	40710.3	1393	39366.5	1658
45	40930.25	1765	36849	1461
46	37103.3	2003	35332.7	1581
47	42784.95	1968	37980	1627
48	42815.9	1486	40800.5	3179
49	45995.3	1817	42270	3250
50	43632.9	2031	39049	1619
51	47874.5	2113	42351	1731
52	41940.4	1409	39315.7	3214
53	42631.7	1824	39705.7	1835
54	47850	1783	43014	1901
55	34912.9	1925	31949.5	3042
56	38373.4	1883	32042	1616
57	36740.9	2041	35574	1548
58	39070.1	1948	33097	3032
59	40254.3	1652	35746	1529
60	43955.4	2134	38085	1794
61	43486.7	1867	40045	3156
62	46305.89	1894	41798	1552
63	46699.1	1858	40897	1741

Table-2 Relaxation: P3 V/s Relaxation: P4

Problem No.	Relaxation: P3	Iterations	Relaxation: P4	Iterations
1	44705	631	46591.1	1133
2	48334.3	601	49962	1360
3	47064.78	645	47362	974
4	52003.4	741	52170.8	1072
5	37825.5	648	41352.6	1163
6	47358.4	657	50130.8	1241
7	50846.3	707	51488.3	940
8	54883.9	692	56048.7	1243
9	52529.6	677	53800.2	1203
10	45079.3	580	46699.02	1499
11	39344.8	679	41184.1	1111
12	46894.9	681	48484	1801
13	44456.2	703	46668.7	1224
14	49681.2	681	50688.9	1158
15	45072.3	681	47320	985
16	45965.4	537	50073.8	993
17	46266.2	713	47670.6	1177
18	43335.3	607	44024.2	1317
19	45460	614	47292.8	1135
20	47528.7	751	48638.4	1460
21	49821.8	664	50235.5	1320
22	46340	764	51056.8	1237
23	40954.6	758	43944.9	1211
24	46155.3	665	47913.5	1351
25	39477.8	581	43542.1	1236
26	37147.2	709	39537	1153
27	51754.7	584	54055.4	1153
28	49966.9	684	50698.3	1124
29	40703.3	853	43323.6	1231
30	56800	555	58956.6	1125
31	41279.7	547	43117.4	1180
32	40137	642	41141.1	1144
33	55989.32	688	57882.3	1223
34	36452.4	983	39572.5	1115
35	47127.4	669	48432	888
36	36682.6	756	41212.2	1223
37	51360	543	54745.2	991
38	44885.9	452	45781.4	1224
39	46525.8	687	48046.5	1200
40	40459.7	589	45030.4	1096
41	42408.3	777	46925.2	837

42	38826.5	708	40302.9	1259
43	44438.2	603	44581.1	1245
44	45373.9	499	46863	1014
45	42072	516	44775.2	1269
46	46843.3	479	48956.5	1078
47	45091.7	664	46649.6	1136
48	50016	652	52103	1045
49	47868.4	714	49644.7	1075
50	48695	678	50477.3	1301
51	45222.7	743	48481.6	1118
52	47741.3	565	48739.8	710
53	34584.4	765	37744.7	1128
54	41236.4	661	45312.7	1533
55	41750.1	726	45006.3	1310
56	40402.8	522	45016.3	1249
57	46132.9	585	48681.6	1249
58	51908.3	506	53667.9	1519
59	51695.2	610	53101.51	1267
60	47174.5	842	48019.3	1097
61	49499.9	671	53795	1201

Table 3 Relaxation: P1 V/s Relaxation: P3

Problem No.	Relaxation: P1	Iterations	Relaxation: P3	Iterations
1	41814	1949	44705	631
2	45832.9	1999	48334.3	601
3	42665.5	2280	47064.78	645
4	47368	1905	52003.4	741
5	35669.9	2053	37825.5	648
6	42208.8	1567	47358.4	657
7	46735.1	1905	50846.3	707
8	48929.8	1529	54883.9	692
9	45182.4	2028	52529.6	677
10	36467.1	1544	38671.1	620
11	41685.2	1356	45079.3	580
12	43404.1	1970	46894.9	681
13	42198.7	1651	44456.2	703
14	44578	1766	49681.2	681
15	41352.9	1735	45072.3	681
16	39547.3	1287	45965.4	537
17	38335.8	1648	46266.2	713
18	38868.5	1659	43335.3	607
19	41212.7	1803	45460	614
20	47660.1	1278	49821.8	664
21	43629.3	1839	46340	764
22	37420	1485	40954.6	758
23	43281.6	2052	46155.3	665
24	35540.6	2111	39477.8	581
25	35571.6	1435	37147.2	709
26	41388.6	2122	43975.9	672
27	40169.5	2008	44516.3	645
28	46888.3	1932	51754.7	584
29	45915.1	1965	49966.9	684
30	36205	1785	40703.3	853
31	53826.6	2385	56800	555
32	36114	2205	41279.7	547
33	37520.1	1762	40137	642
34	51172.7	2073	55989.32	688
35	34846.7	1312	36452.4	983
36	41304.7	1510	47127.4	669
37	36149.7	1955	36682.6	756
38	47329.2	1971	51360	543
39	39667.1	2075	44885.9	452
40	39841.3	1967	45605.34	493
41	38522.13	1519	42408.3	777
42	35492	2087	38826.5	708

43	40710.3	1393	44438.2	603
44	40930.25	1765	45373.9	499
45	37103.3	2003	42072	516
46	42784.95	1968	46843.3	479
47	42815.9	1486	45091.7	664
48	45995.3	1817	50016	652
49	43632.9	2031	47868.4	714
50	41940.4	1409	45222.7	743
51	42631.7	1824	47741.3	565
52	47850	1783	52129.1	705
53	34912.9	1925	34584.4	765
54	38373.4	1883	41236.4	661
55	36740.9	2041	41750.1	726
56	39070.1	1948	40402.8	522
57	40254.3	1652	46132.9	585
58	43955.4	2134	51908.3	506
59	42189.31	1561	51695.2	610
60	43486.7	1867	47174.5	842
61	46699.1	1858	49499.9	671

APPENDIX (F)

The details about the problems, generated through JAVA program

Value of fixed cost varies from min =500 to max =4000

i.e. F(i) =500 to 4000

Demand from market k, D(k) varies from min =1 to max =100

in fraction i.e. for d(k) =1

and d(k) =D(k)/ D(k)

cost for transporting unit quantity from one plant to market varies from min =1 to max = 100 and when this cost is infinite then the cost(i, k) =100000000.

C(i, k) for fulfilling the demand of all the markets from plant i varies from

Dk*cost(i, k)

APPENDIX (G)

Table 1 Results From 70 x 70 Problem Set

		SRS				WRS(OLD)			WRS(NEW)		
		Without Big 'M'		*With Big 'M'*		*Without Big 'M'*		*With Big 'M'*	*Without Big 'M'*		*With Big 'M'*
	Problem	*Value*	*Iterations*	*Value*	*Iteration*	*Value*	*Iterations*	*Value*	*Value*	*Iterations*	*Value*
70 x 70	1	55699.0	21535	55699.0	12017	33459.9	751	33459.9	33627.5	852	33627.4
	2	67022.5	20373	67022.5	10648	42145.4	839	42145.4	42113.4	776	42113.4
	3	67687.0	10788	67687.0	10689	42986.8	1191	42986.8	42986.1	945	42986.1
	4	73906.3	20655	73906.3	21848	44504.9	1611	44504.9	44550.4	869	44550.4
	5	60704.0	11098	60704.0	11415	38469.2	724	38469.3	38662.6	871	38662.6
	6	62165.0	21147	62165.0	11470	37848.1	734	37848.1	38047.0	785	38047.0
	7	64940.0	11273	64940.0	15617	38212.7	1302	38212.7	38221.2	806	38221.2
	8	67908.0	20124	67908.0	20703	42742.6	833	42742.6	42715.0	979	42715.0
	9	70353.5	10191	70353.5	20490	44132.6	741	44132.7	44196.8	947	44196.8
	10	69006.5	20403	69006.5	21187	38276.0	867	38276.0	38230.2	956	38230.2
	11	61585.0	11151	61585.0	22306	42529.4	1189	41747.7	42509.9	906	41663.0
	12	71307.0	21062	71307.0	20469	41612.6	1138	41642.6	41727.1	812	41727.0
	13	71239.3	21910	71239.3	21151	39265.6	874	39265.6	39348.9	824	39348.9
	14	68691.7	11133	68691.7	22165	39578.7	808	39385.5	39385.5	1000	39578.7
	15	65768.7	21601	65768.7	10762	38856.3	725	38856.3	39057.6	895	39057.6

Here Value of M=1000000

Table 2 Results From 60 x 60 Problem Set

		SRS				WRS(OLD)			WRS(NEW)		
		Without Big 'M'		*With Big 'M'*		*Without Big 'M'*		*With Big 'M'*	*Without Big 'M'*		*With Big 'M'*
	Problem	*Value*	*Iterations*	*Value*	*Iteration*	*Value*	*Iterations*	*Value*	*Value*	*Iterations*	*Value*
60 x 60	1	62874.0	14278	62874.0	15884	38330.3	601	38330.2890	38438.6	593	38438.6
	2	63185.0	15083	63185.0	15477	36228.3	997	36228.3125	36321.4	618	36321.4
	3	61279.7	15801	61279.7	16170	38085.9	1134	38085.9250	38183.4	716	38183.4
	4	59137.5	8331	59137.5	8049	35121	1036	35121.0000	35124.8	952	35124.8
	5	55110.0	7774	55110.0	8260	36169.6	1104	36169.5930	36223.8	772	36223.8
	6	57388.0	7836	57388.0	15079	36113.9	965	36113.8710	36126.3	757	36126.3
	7	60157.0	8040	60157.0	14967	37266.9	779	37266.8515	37199.4	728	37199.4
	8	60965.0	7884	60965.0	15526	35876.9	538	35876.8750	35964.7	629	35964.7
	9	61464.0	15792	61464.0	16031	35876.9	538	38258.8500	35964.7	639	38324.6
	10	55549.0	7631	55549	15348	38258.9	620	32131.3400	38324.6	679	32297.1
	11	53897.0	7799	53897.0	16074	32131.3	716	34085.9258	32297.1	654	34139.8
	12	55913.0	7692	55913.0	15047	34085.9	642	34635.1290	34139.8	788	34591.1
	13	62189.3	8118	62189.3	7971	37222.8	887	37228.3900	37365.9	658	37365.9
	14	61301.0	14962	61301.0	15090	39417.1	685	39417.1523	39405.8	756	39405.8
	15	52977.0	7788	52977.0	15210	30876.5	1199	30876.4512	30846.7	723	30846.7

Here Value of M=1000000

REFERENCES

1. Bilde,O. and Krarup,J. "Sharp lower bounds and efficient algorithms for the simple plant location problem", *Annals of Discrete Mathematics*, 1(1977), 79-97.

2. Balinski, M.L. and P.Wolfe, "On Benders decomposition and a plant location problem, working paper ARO-27", *Mathematica, Princeton, NJ* (1963).

3. Chhajed, B., Francis, G.I. and Lowe, A.C. "Contributions of operations research to location analysis", *Location Science*, 1(1993), 263-287.

4. Cooper, L. "Location - allocation problem", *Operations Research*, 11(1963), 331-343.

5. Cornuejols, G. Fisher, M.L. and Nemhauser, G.L. "Location of Bank accounts to optimise float: An analytic study of exact and approximate algorithms", *Management Science*, 23(1977), 789-810.

6. Cornuejols, G, Nemhauser, G.L. and Wolsey, L.A. "The uncapacitated facility location problem", *MSSR 493, Graduate School of Industrial Administration, CMU* (1983).

7. Cornuejols, G., Sridharan, R. and Thizy, J.M., "A comparison of heuristics and relaxations for the Capacitated Plant Location Problem", *European Journal of Operation Research,* 50(1991)280-287.

8. Effroymson, M. and Ray, T. "A branch and bound algorithm for plant location", *Operations Research*, 14(1966), 361-368.

9. Erlenkotter, D. "A dual based procedure for uncapacitied facility location", *Operations Research*, 26(1978), 992-1009.

10. Feldman, E, Leher, F.A. and Ray T.L. "Warehouse location under continuous economies of scale", *Management Science*, 2(1966),670-684.

11. Galvao, R. "The use of lagrangian relaxation in the solution to uncapacitated facility location problems", *Location Science*, 1(1993), 57-59.

12. Jacobson, S.K. "Heuristics for the capacitated plant location model", *European Journal of Operations Research*, 12(1983), 253-261.

13. Khumawallah, B.M. "An efficient heuristic procedure for capacitied warehouse location problem", *Naval Research Logistics Quarterly*, 21(4)(1974), 609-623.

14. Krarup, J. and Pruzan, P.M. "The simple plant location problem: Survey and synthesis", *European Journal of Operations Research*, 12(1983), 36-81.

15. Kuehn, A.A and Hamburger, M.J. "A heuristic program for locating warehouses", *Management Science*, 9(1963), 643-666.

16. Manne, A.S. "Plant location under economics-of-scale-Decentralization and computation", *Management science*, 11 (1964),213-235.

17. Sharma, R.R.K. and Berry, V. "Developing Different Formulations Of SSCWLP & Empirically Establishing Relative Strengths Of Many Of Its Relaxations", *M.Tech. Thesis*, 2003, *IME Dept., IIT Kanpur.*

18. Sharma, R.R.K. and Muralidhar, A.B.V.N. "A New Procedure For Solving Simple Plant Location Problem", *M.Tech Thesis,* 2000, *IME Dept., IIT Kanpur.*

19. Sharma R.R.K. and Sharma K.D. "A new dual based procedure for the transportation problem", *European Journal of Operational Research,* 122(3)(2000),37-48.